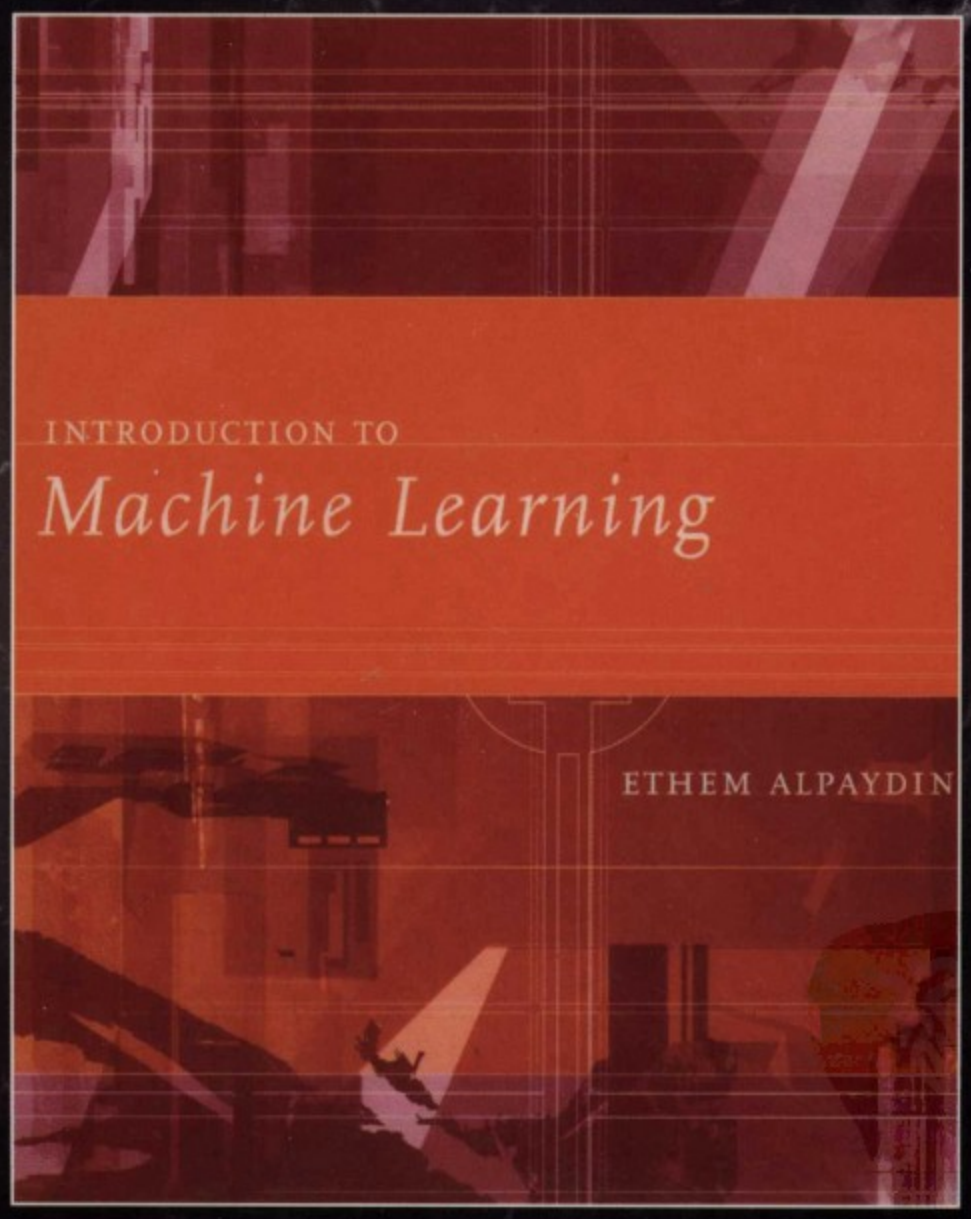
HZ BOOKS
华章教育

计 算 机 科 学 丛 书

机器学习导论

（土耳其）Ethem Alpaydın 著　范明 咎红英 牛常勇 译



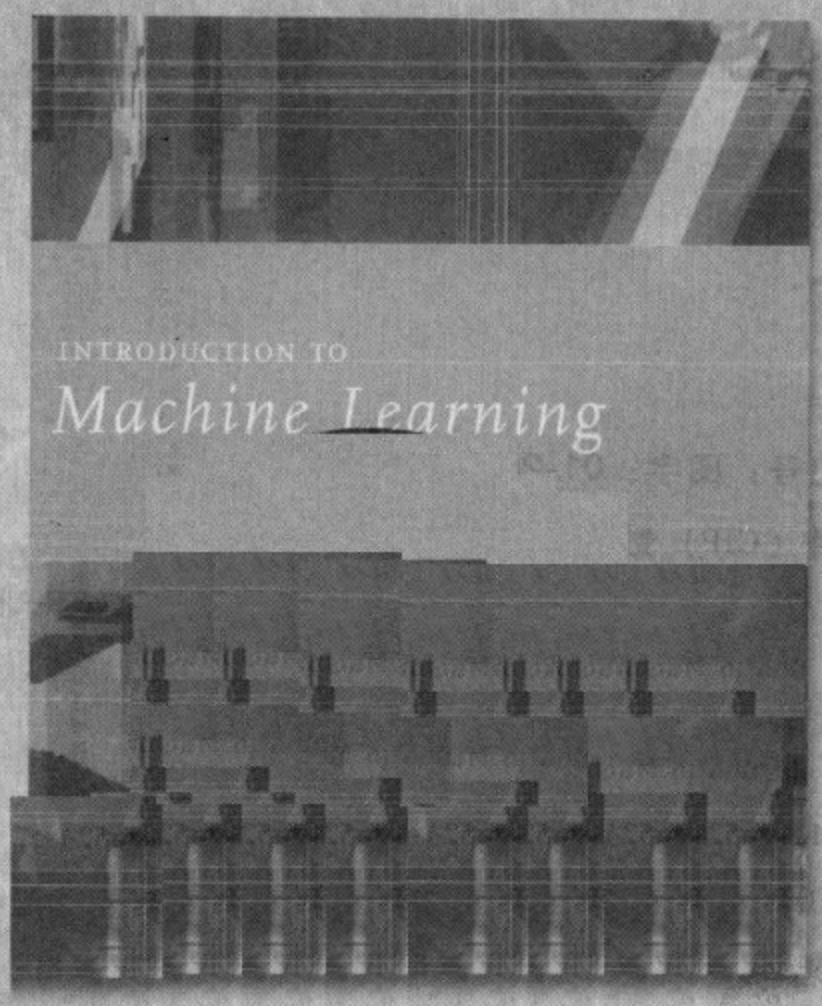
Introduction to Machine Learning

机 械 工 业 出 版 社
China Machine Press

计 算 机 科 学 丛 书

机器学习导论

（土耳其）Ethem Alpaydın 著　范明 咎红英 牛常勇 译



Introduction to Machine Learning

机 械 工 业 出 版 社
China Machine Press

本书讨论了机器学习在统计学、模式识别、神经网络、人工智能、信号处理等不同领域的应用。其中涵盖的内容比较全面，且易于学习和掌握。主要内容包括：监督学习、贝叶斯决策理论、参数方法、多元方法、维度归约、非参数方法、决策树、线性判别式、多层感知器、隐马尔可夫模型、组合多学习器以及增强学习等。可作为高等院校计算机相关专业高年级本科生和研究生的教材，也可供研究机器学习方法的技术人员参考。

本书版权登记号：图字：01-2006-5437

图书在版编目（CIP）数据

机器学习导论/（土）阿培丁（Alpaydın, E.）著；范明等译. —北京：机械工业出版社，2009.6

（计算机科学丛书）

书名原文：Introduction to Machine Learning

ISBN　978-7-111-26524-5

Ⅰ.机…　Ⅱ.①阿…②范…　Ⅲ.机器学习　Ⅳ.TP181

中国版本图书馆 CIP 数据核字（2009）第 031371 号

机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码　100037）

责任编辑：李俊竹

北京慧美印刷有限公司印刷

2009 年 6 月第 1 版第 1 次印刷

184mm × 260mm · 18 印张

标准书号：ISBN 978-7-111-26524-5

定价：39.00 元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章分社较早意识到“出版要为教育服务”。自1998年开始，华章分社就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出Andrew S. Tanenbaum，Bjarne Stroustrup，Brain W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章分社欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com

电子邮件：hzjsj@hzbook.com

联系电话：（010）88379604

联系地址：北京市西城区百万庄南街1号

邮政编码：100037

译者序

自从有计算机以来，人们就希望计算机能够学习。然而，机器学习真正取得实质性进展，能够成功地解决一些实际问题，并最终成为一个学科分支还是近20余年的事。

对于许多问题，我们的前人和先行者已经知道如何求解。例如，欧几里德告诉我们可以用辗转相除法求两个整数的最大公约数；Dijkstra告诉我们如何有效地求两点之间的最短路径；Hoare向我们展示了怎样将杂乱无章的对象快速排序……对于这些问题，我们清楚地知道求解步骤。因此，让计算机求解这些问题只需要设计算法和数据结构、进行编程，而不需要让计算机学习。

还有一些事情，人们可以轻而易举地做好，但是却无法解释清楚我们是如何做的。例如，尽管桌子千差万别、用途各异，但是我们一眼就能看出某个物体是否是桌子；尽管不同的人的手写阿拉伯数字大小不一、笔画粗细不同，但是我们还是可以轻易识别一个数字是不是8；尽管声音时大时小、有时可能还有点沙哑，但是我们还是可以不费力气地听出熟人的声音。诸如此类的例子不胜枚举。对于这些问题，我们不知道求解步骤。因此，让计算机来做这些事就需要让计算机学习。

我们知道桌子不是木材和各种材料的随机堆砌，手写数字不是像素的随机分布，熟人的声音也不是各种声波的随机混合。现实世界总是有规律的。机器学习正是从已知实例中自动发现规律，建立对未知实例的预测模型；根据经验不断提高，不断改进预测性能。

这是关于机器学习这一主题全面论述的教科书，适合作为高等院校计算机相关专业高年级本科生和研究生机器学习入门课程的教材。该书涵盖了监督学习、贝叶斯决策理论、参数方法、多元方法、维度归约、聚类、非参数方法、决策树、线性判别式、多层感知器、局部模型、隐马尔可夫模型、分类算法评估和比较、组合多学习器以及增强学习。作者对来自统计学、模式识别、神经网络、人工智能、信号处理、控制和数据挖掘等不同领域的机器学习问题和学习方法进行了统一论述。

现在，学习的本质还不十分清楚。然而，关于学习的理论认识已开始逐步形成，已经建立起来的一些机器学习方法已经成功地解决了许多实际问题。我们能够从这本书中学习机器学习，发现机器学习的新方法，不断提高对学习本质的认识。

全书共分16章和一个附录。昝红英翻译了第1~6章，牛常勇翻译了第13~16章，范明翻译了其余部分，并对全书译文进行了修改和最后定稿。

本书的翻译得到了原作者Ethem Alpaydın教授的支持。Ethem教授不仅为中文版写序，而且还耐心地解释了我们的一些疑问。在此，我们向Ethem教授表示感谢。

译文中的错误和不当之处，敬请读者朋友指正。意见和建议请发至mfan@zzu.edu.cn，我们不胜感激。

译　者

2009年春于郑州大学

致 谢

获得好想法的途径是与有才干的人一起工作，与他们一起工作也是一种乐趣。Boğaziçi 大学计算机工程系是一个极好的工作场所，在我写这本书时，我的同事们为我提供了我所需要的所有支持。我也要感谢我过去和现在的学生，在他们身上，我实际检验了现在写进这本书中的内容。

在写本书时，我得到了土耳其科学院青年科学家奖励计划的资助（EATÜBA-GEBİP/2001-1-1）。

我特别感谢 Michael Jordan。对于他多年来的支持和最近对本书的支持，我深表感谢。他针对本书大体组织和第 1 章所给出的建议在内容和形式上都大大改进了本书。Taner Bilgiç、Vladimir Cherkassky、Tom Dietterich、Fikret Gürgen、Olcay Taner Yıldız 和 MIT 出版社的未留名审稿人也部分阅读了本书，并提供了非常宝贵的反馈。我希望他们在注意到我采纳了他们的建议但却没有特别致谢时，能够体会到我的感激之情。当然，书中的错误和不足应当由我个人负责。

我的父母信任我，我感谢他们永恒的爱和支持。无论我何时需要，Sema Oktuğ 总在身边，我将永远感激她的友谊。我还要感谢 Hakan Ünlü，在过去的几年中，我们无数次讨论了涉及生活、宇宙和万事万物的众多主题。

本书使用 Chris Manning 准备的 LATEX 宏排版，对此我很感谢他。我要感谢 MIT 出版社的编辑们，以及 Bob Prior、Valerie Geary、Kathleen Caruso、Sharon Deacon Warne、Erica Schultz 和 Emily Gutheinz，感谢他们在本书完成期间的不断支持和帮助。

符号表

x	标量值
$\boldsymbol{x}$	向量
$\mathbf{X}$	矩阵
$\boldsymbol{x}^T$	转置
$\mathbf{X}^{-1}$	逆矩阵
X	随机变量
$P(X)$	概率质量函数，X 是离散的
$p(X)$	概率质量函数，X 是连续的
$P(X \mid Y)$	给定 Y，X 的条件概率
$E[X]$	随机变量 X 的期望值
$\mathrm{Var}(X)$	X 的方差
$\mathrm{Cov}(X, Y)$	X 和 Y 的协方差
$\mathrm{Corr}(X, Y)$	X 和 Y 的相关性
μ	均值
σ^2	方差
Σ	协方差矩阵
m	均值的估计
s^2	方差的估计
$\mathbf{S}$	协方差矩阵的估计
$\mathcal{N}(\mu, \sigma^2)$	一元正态分布，均值为 μ，方差为 σ^2
$\mathcal{Z}$	单位正态分布：$\mathcal{N}(0, 1)$
$\mathcal{N}_d(\mu, \Sigma)$	d-变量正态分布，均值向量为 μ，协方差矩阵为 Σ
x	输入
d	输入数：输入的维度
y	输出
r	要求的输出
K	输出数(类)
N	训练实例数
z	隐藏的值，内蕴维，潜在因子
k	隐藏维数，潜在因子数

C_i	类 i
χ	训练样本
$\{x^t\}_{t=1}^N$	x 的集合，上标 t 遍取 1 到 N
$\{x^t, r^t\}_t$	上标为 t 的输入和期望输出的有序对的集合
$g(x \mid \theta)$	x 的函数，其定义依赖于参数集 θ
$\arg\max_\theta g(x \mid \theta)$	参数 θ，g 关于它取最大值
$\arg\min_\theta g(x \mid \theta)$	参数 θ，g 关于它取最小值
$E(\theta \mid \chi)$	样本 χ 上具有参数 θ 的误差函数
$l(\theta \mid \chi)$	样本 χ 上具有参数 θ 的似然函数
$\mathcal{L}(\theta \mid \chi)$	样本 χ 上具有参数 θ 的对数似然函数
$1(c)$	如果 c 为真，则值为 1，否则为 0
$\#\{c\}$	c 为真的元素数目
δ_{ij}	Kronecker δ：如果 $i=j$，取 1，否则取 0

前言

机器学习使用实例数据或过去的经验训练计算机，以优化性能标准。当人们不能直接编写计算机程序解决给定的问题，而是需要借助于实例数据或经验时，就需要学习。一种需要学习的情况是人们没有专门技术，或者不能解释他们的专门技术。以语音识别，即将声学语音信号转换成 ASCII 文本为例。看上去我们可以毫无困难地做这件事，但是我们却不能解释我们是如何做的。由于年龄、性别或口音的差异，不同的人读相同的词发音却不同。在机器学习中，这个问题的解决方法是从不同的人那里收集大量发音样本，并学习将它们映射到词。

另一种需要学习的情况是要解决的问题随时间变化或依赖于特定的环境。我们希望有一个能够自动适应环境的通用系统，而不是为每个特定的环境编写一个不同的程序。以计算机网络上的包传递为例。最大化服务质量的、从源地到目的地的路径随网络流量的改变而改变。学习路由程序能够通过监视网络流量自动调整到最佳路径。另一个例子是智能用户界面，它能够自动适应用户的生物特征，即用户的口音、笔迹、工作习惯等。

机器学习在各个领域都有许多成功的应用：已经有了识别语音和笔迹的商用系统。零售商分析他们过去的销售数据，了解顾客行为，以便改善顾客关系管理。金融机构分析过去的交易，以便预测顾客的信用风险。机器人学习优化它们的行为，以便使用最少的资源来完成任务。在生物信息学方面，使用计算机不仅可以分析海量数据，而且还可以提取知识。这些只是我们（即你和我）将在本书讨论的应用的一部分。我们只能想象一下可使用机器学习实现的未来应用：可以在不同的路况、不同的天气条件下自己行驶的汽车，可以实时翻译外语的电话，可以在新环境（例如另一个星球的表面）航行的自动化机器人。机器学习的确是一个令人激动的研究领域！

本书讨论的许多方法都源于各种领域：统计学、模式识别、神经网络、人工智能、信号处理、控制和数据挖掘。过去，这些不同领域的研究遵循不同的途径，侧重点也不同。本书旨在把它们组合在一起，给出问题的统一处理并提供它们的解。

本书是一本入门教材，用于高年级本科生和研究生的机器学习课程，以及在业界工作、对这些方法的应用感兴趣的工程技术人员。预备知识是计算机程序设计、概率论、微积分和线性代数方面的课程。本书的目标是充分解释所有的学习算法，使得从本书给出的方程到计算机程序只是一小步。为了使这一任务更容易完成，对于某些情况，我们给出了算法的伪代码。

适当选取一些章节，本书可用作一学期的课程。再额外讨论一些研究论文的话，本书也可以作为两学期的课程，这时每章后的参考文献将很有用。

本书网页为 http://www.cmpe.boun.edu.tr/~ethem/i2ml/，我将在那里提供一些与本书有关的信息，如勘误表。我真诚地欢迎你将你的反馈意见发到我的邮箱：alpaydin@boun.edu.tr。

我非常喜欢写这本书；希望你能喜欢读它。

中文版序

机器学习领域在理论和应用两方面都发展迅速。无论是学术界还是产业界，人们都对能够通过实例学习的计算机程序表现出了极大的兴趣，并且所有国家都是如此。因此，看到本书的中文版出版我特别高兴，另外，我感谢范明教授为翻译本书所做出的努力，他在此之前翻译了几本统计学和数据挖掘的名著。我希望本书的读者能觉得它有益处，并且就像我乐于写它一样乐于阅读它。

Ethem Alpaydın
于伊斯坦布尔 Boğaziçi 大学
2008. 8

Preface of the Chinese Edition

The field of machine learning is developing rapidly both in theory and applications. There is great interest in computer programs which can learn from examples, both in academia and industry, and this is true for all countries. It therefore gives me great pleasure to see the Chinese language edition of my book in print, and for the effort in dong the translation, I thank Professor Fan who previously have translated several well-known texts on statistics and data mining. I hope that the readers of my book will find it beneficial and enjoy reading it as much as Ienjoyed writing it.

Ethem Alpaydın
Boğaziçi University, Istanbul
August 2008

目　录

第1章 绪 论

1.1 什么是机器学习

随着计算机技术的发展，我们现在已经拥有存储和处理海量数据以及通过计算机网络从远程站点访问数据的能力。目前大多数的数据存取设备都是数字设备，记录的数据也很可靠。以一家连锁超市为例，它拥有遍布全国各地的数百家分店，并且在为数百万顾客提供数千种商品的零售服务。销售点的终端设备记录每笔交易的详细资料，包括日期、顾客识别码、购买商品和数量、消费总额等。这是典型的每日几个 G 字节的数据。只有分析这些数据，并且将它转换为可以利用的信息时，这些存储的数据才能变得有用，例如做预测。

我们不能确切地知道哪些人比较倾向于购买哪种特定的商品，也不知道应该向喜欢读海明威作品的人推荐哪位作者。如果我们知道，我们就不需要任何数据分析；我们只管供货并记录下编码就可以了。但是，正因为我们不知道，所以才只能收集数据，并期望从数据中提取这些问题或相似问题的答案。

我们确信存在某种过程，可以解释我们所观测到的数据。尽管我们不清楚数据产生过程（例如顾客行为）的细节，但是，我们知道数据产生不是完全随机的。人们并不是去超市随机购买商品。当人们买啤酒时，也会买薯片；夏天买冰淇淋，而冬天则为 Glühwein[⊖] 买香料。数据中存在确定的模式。

我们也许不能够完全识别该过程，但是我们相信，我们能够构造一个好的并且有用的近似（good and useful approximation）。尽管这样的近似还不可能解释一切，但其仍然可以解释数据的某些部分。我们相信，尽管识别全部过程也许是不可能的，但是我们仍然能够发现某些模式或规律。这正是机器学习的定位。这些模式可以帮助我们理解该过程，或者我们可以使用这些模式进行预测：假定将来，至少是不远的将来，情况不会与收集样本数据时有很大的不同，则未来的预测也将有望是正确的。

机器学习方法在大型数据库中的应用称为*数据挖掘*（data mining）。类似的情况如大量的金属氧化物以及原料从矿山中开采出来，处理后产生少量非常珍贵的物质。同样地，在数据挖掘中，需要处理大量的数据以构建简单有用的模型，例如具有高精度的预测模型。数据挖掘的应用领域非常广泛：除零售业以外，在金融业，银行分析他们的历史数据，构建用于信用分析、诈骗检测、股票市场等方面的应用模型；在制造业，学习模型可以用于优化、控制以及故障检测等；在医学领域，学习程序可以用于医疗诊断等；在电信领域，通话模式的分析可用于网络优化和提高服务质量；在科学研究领域，比如物理学、天文学以及生物学的大

⊖ Glühwein 是一种温热、有点甜味、加香料的葡萄酒。圣诞节期间，在欧洲很受欢迎。——译者注

量数据只有用计算机才可能得到足够快的分析。万维网(World Wide Web)是巨大的，并且在不断地增长，因此在万维网上检索相关信息不可能依靠人工完成。

然而，机器学习不仅仅是数据库方面的问题，它也是人工智能的组成部分。为了智能化，处于变化环境中的系统必须具备学习的能力。如果系统能够学习并且适应这些变化，那么系统的设计者就不必预见所有的情况，并为它们提供解决方案了。

机器学习还可以帮助我们解决视觉、语音识别以及机器人方面的许多问题。以人脸识别问题为例：我们做这件事毫不费力；即使姿势、光线、发型等不同，我们每天还是可以通过看真实的面孔或其照片来认出我们的家人和朋友。但是我们做这件事是下意识的，而且无法解释我们是如何做的。因为我们不能够解释我们所具备的这种技能，我们也就不可能编写相应的计算机程序。但是我们知道，脸部图像并非只是像素点的随机组合；人脸是有结构的、
2 对称的。脸上有眼睛、鼻子和嘴巴，并且它们都位于脸的特定部位。每个人的脸都有各自的眼睛、鼻子和嘴巴的特定组合模式。通过分析一个人脸部图像的多个样本，学习程序可以捕捉到那个人特有的模式，然后在所给的图像中检测这种模式，从而进行辨认。这就是模式识别(pattern recognition)的一个例子。

机器学习使用实例数据或过去的经验训练计算机，以优化某种性能标准。我们有依赖于某些参数的模型，而学习就是执行计算机程序，利用训练数据或以往经验来优化该模型的参数的。模型可以是预测性的(predictive)，用于未来的预测，或者是描述性的(descriptive)，用于从数据中获取知识，也可以二者兼备。

机器学习在构建数学模型时利用了统计学理论，因为其核心任务就是从样本中推理。计算机科学的角色是双重的：第一，在训练时，我们需要求解优化问题以及存储和处理通常所面对的海量数据的高效算法。第二，一旦学习得到了一个模型，它的表示和用于推理的算法解也必须是高效的。在特定的应用中，学习或推理算法的效率，即它的空间复杂度和时间复杂度，可能与其预测精确度同样重要。

现在，让我们更详细地讨论一些应用领域的例子，以进一步深入了解机器学习的类型和用途。

1.2 机器学习的应用实例

1.2.1 学习关联性

在零售业，例如超市连锁店，机器学习的一个应用是购物篮分析(basket analysis)。它的任务是发现顾客所购商品之间的关联性：如果人们在购买商品 X 时也通常购买商品 Y，而有一名顾客购买了商品 X 却没有购买商品 Y，则他(或她)即是商品 Y 的潜在顾客。一旦我们发现这类顾客，我们就能针对他们实行打包销售策略。

为发现关联规则(association rule)，我们对学习形如 $P(Y \mid X)$ 的条件概率感兴趣，其中
3 X 是我们知道的顾客已经购买的商品或商品集，Y 表示在条件 X 下可能购买的商品。

假定考察已有的数据，计算得到 $P(\text{chips} \mid \text{beer}) = 0.7$，那么我们就可以定义规则：购买啤酒(beer)的顾客中有70%的人也买了薯片(chips)。

我们也许想要区分不同的顾客。针对这个问题，我们需要估计 $P(Y \mid X, D)$，其中 D 是

顾客的一组属性，如性别、年龄、婚姻状况等，这里假定我们已经得到了这些属性信息。如果是考虑书店而不是超市的销售问题，商品就可能是书或作者等。对于 Web 门户网站入口问题，项对应着到 Web 网页的链接，而我们可以估计用户可能点击的链接，并利用这些信息来预先下载这些网页，以取得更快的网页存取速度。

1.2.2　分类

信贷是金融机构(例如银行)借出的一笔钱，需要连本带息偿还，通常是分期偿还。对银行来说，重要的是能够提前预测贷款风险。这种风险是客户不履行义务和不全额还款的可能性。既要确保银行获利，又要确保不会因提供超出客户财力的贷款而给客户带来不便。

在信用评分(credit scoring)(Hand 1998)中，银行要计算在给定信贷额度和客户信息情况下的风险。客户信息包括我们已经获取的数据以及与计算客户财力相关的数据，即收入、存款、担保、职业、年龄、以往经济记录等。银行有以往贷款的记录，包括客户数据以及贷款是否偿还。通过这类特定的申请数据，我们可以推断出一般规则，表示客户属性及其风险性的关联性。也就是说，机器学习系统用一个模型来拟合过去的数据，以便能够对新的申请计算风险，从而决定接受或拒绝该项申请。

这是一个分类(classification)问题的例子，这里有两个类：低风险客户和高风险客户。客户信息作为分类器的输入(input)，分类器的任务是将输入指派到其中的一个类。

利用以往数据进行训练后，学习得到的规则可能具有如下形式

$$\text{IF income} > \theta_1 \text{ AND savings} > \theta_2 \text{ THEN low-risk ELSE high-risk}$$

其中 θ_1 和 θ_2 是合适的值(参见图 1-1)。这是判别式(discriminant)的一个例子，它是将不同类的样本分开的函数。

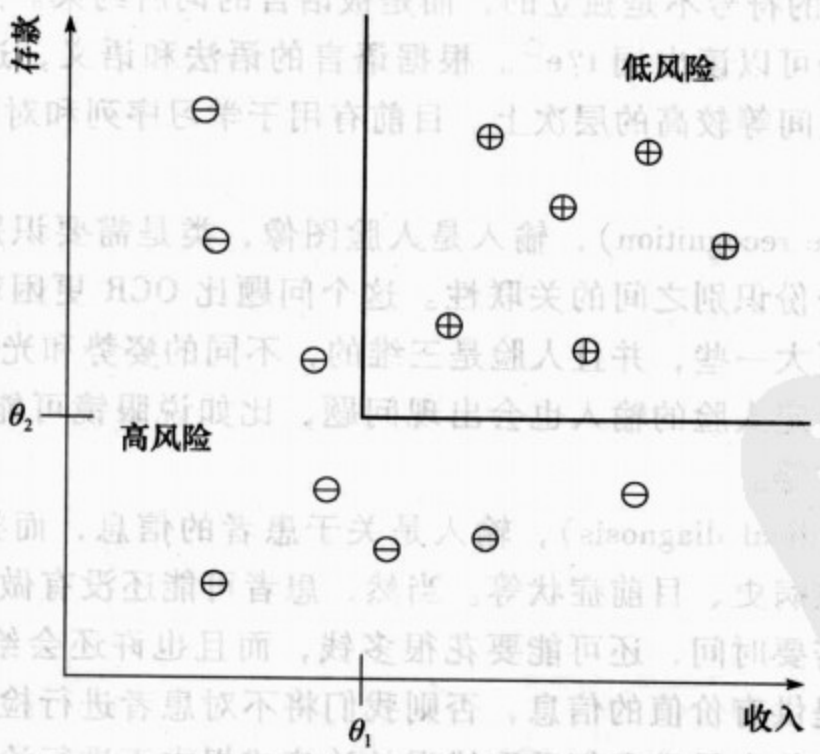


图 1-1　训练数据集示例，其中每个圆圈对应一个数据实例，输入值在对应的坐标上，符号则指示着类别。为简单起见，输入只包括客户的收入(income)和存款(savings)两种属性，两个类分别为低风险(“+”)和高风险(“-”)。图中还显示了分隔两类样本的判别式样例

有了这样的规则，其主要用途就是预测(prediction)：一旦我们拥有拟合以往数据的规则，如果未来与过去类似，那么我们就能够对新的实例做出正确的预测。如果给定一个新的具有特定收入(income)和存款(savings)的申请，我们就可以很容易地判断出它是低风险(low-risk)还是高风险(high-risk)了。

在某些情况下，我们可能不希望做0/1(低风险/高风险)类型的判断，而是希望计算
5 一个概率值 $P(Y \mid X)$，其中 X 是顾客属性，Y 是0或1，分别表示低风险和高风险。从这个角度来看，我们可以将分类看作学习从 X 到 Y 的关联性。于是，给定 $X=x$，如果有 $P(Y=1 \mid X=x)=0.8$，则我们就说该客户为高风险的可能性有80%，或者等价地说该客户为低风险的可能性有20%。然后，我们可以根据可能的收益和损失来决定接受或拒绝这笔贷款业务。

机器学习在模式识别(pattern recognition)方面有很多的应用。其中之一是光学字符识别(optical character recognition，OCR)，即从字符图像识别字符编码。这是一个多类问题的例子，类与我们想要识别的字符一样多。特别有趣的是手写体字符的识别问题。人们有不同的书写风格；字体有大有小，倾斜角度不同，还有用钢笔或用铅笔之别，所以同一个字符可能会有许多种可能的图像。尽管书写是人类的发明创造，但是我们还没有像人类读者一样准确的系统。我们没有字符“A”的形式化描述，涵盖所有“A”而不涵盖任何非“A”。没有这种形式化描述，我们就要从书写者那里取样，从这些实例中学习关于“A”的定义。然而，尽管我们不知道是什么因素使得一个图像被识别为“A”，但是我们确信所有这些不同的“A”的图像都具有某些共同的特征，这正是我们希望从实例中提取的。我们知道，图像不只是随机点的集合，它是笔画的集合，并且是有规律的，通过学习程序我们能够捕获这些规律。

阅读文本时，我们能够利用的一个因素是人类语言的冗余性。词是字符的序列(sequence)，并且相继的符号不是独立的，而是被语言的词所约束。这有好处，即便有一个符号不能识别，我们仍可以读出词 t?e[⊖]。根据语言的语法和语义，这种上下文的依赖性还可能出现在词和句子之间等较高的层次上。目前有用于学习序列和对这种依赖性建模的机器学习算法。

对于人脸识别(face recognition)，输入是人脸图像，类是需要识别的人，并且学习程序应当学习人脸图像与身份识别之间的关联性。这个问题比OCR更困难，原因是人脸会有更多的类，输入图像也更大一些，并且人脸是三维的，不同的姿势和光线等都会导致图像的显
6 著变化。另外，对于特定人脸的输入也会出现问题，比如说眼镜可能会把眼睛和眉毛遮住，胡子可能会把下巴盖住等。

对于医学诊断(medical diagnosis)，输入是关于患者的信息，而类是疾病。输入包括患者的年龄、性别、既往病史、目前症状等。当然，患者可能还没有做过某些检查，因此这些输入将会缺失。检查需要时间，还可能要花很多钱，而且也许还会给患者带来不便。因此，除非我们确信检查将提供有价值的信息，否则我们将不对患者进行检查。在医学诊断的情况下，错误的诊断结果可能会导致我们采取错误的治疗或根本不进行治疗。在不能确信诊断结

⊖ 这里，“?”表示不能识别的符号。——译者注

果的情况下，分类器最好还是放弃判定，而等待医学专家来做决断。

在语音识别(speech recognition)，输入是语音，类是可以读出的词汇。这里要学习的是从语音信号到某种语言的词汇的关联性。由于年龄、性别或口音方面的差异，不同的人对于相同词汇的读音不同，这使得语音识别问题相当困难。语音识别的另一个特点是其输入信号是时态的(temporal)，词汇作为音素的序列实时读出，而且有些词汇的读音会较长一些。一种语音识别的新方法涉及利用照相机记录口唇动作，作为语音识别的补充信息源。这需要传感器融合(sensor fusion)技术，集成来自不同形态的输入，即集成声音和视频信号。

从数据中学习规则也为知识抽取(knowledge extraction)提供了可能性。规则是一种解释数据的简单模型，而观察该模型我们能得到潜在数据处理的解释。例如，一旦我们学会了区分低风险客户和高风险客户的判别式，我们就拥有了关于低风险客户特性的知识。然后，我们就能够利用这些知识，通过比如广告等方式，更有效地争取那些潜在的低风险客户。

机器学习还可以进行压缩(compression)。用规则拟合数据，我们能得到比数据更简单的解释，需要的存储空间更少，处理所需要的计算更少。例如，一旦你掌握了加法规则，你就不必记忆每对可能数字的和是多少。

机器学习的另一种用途是离群点检测(outlier detection)，即发现那些不遵守规则的例外实例。在这种情况下，学习规则之后，我们感兴趣的不是规则，而是规则未能覆盖的例外，它们可能暗示出我们需要注意的异常，如诈骗。

7

1.2.3 回归

假设我们想要一个能够预测二手车价格的系统。该系统的输入是我们认为会影响到车价的属性信息：品牌、车龄、发动机性能、里程以及其他信息。输出是车的价格。这种输出为数值的问题是回归(regression)问题。

设 x 表示车的属性，y 表示车的价格。调查一下以往的交易情况，我们能够收集训练数据，而机器学习程序用一个函数拟合这些数据来学习 x 的函数 y。图 1-2 给出了一个例子，其中对于 w 和 w_0 的合适值，拟合函数具有以下形式：

$$y = wx + w_0$$

回归和分类均为监督学习(supervised learning)问题，其中输入 x 和输出 y 给定，任务是学习从输入到输出的映射。机器学习的方法是，先假定某个依赖于一组参数的模型：

$$y = g(x \mid \theta)$$

其中，$g(\cdot)$是模型，θ 是模型的参数。对于回归，y 是数值；对于分类，y 是类编码(如 0/1)。$g(\cdot)$为回归函数，或者(对于分类)是将不同类的实例分开的判别式函数。机器学习程序优化参数 θ，使得逼近误差最小，也就是说，我们的估计要尽可能地接近训练集中给定的正确值。例如，图 1-2 所示的模型是线性的，w 和 w_0 是为最佳拟合训练数据优化的参数。在线性模型限制过强的情况下，我们可以利用比如二次函数：

$$y = w_2x^2 + w_1x + w_0$$

或更高阶的多项式，或其他非线性函数，为最佳拟合优化它们的参数。

8

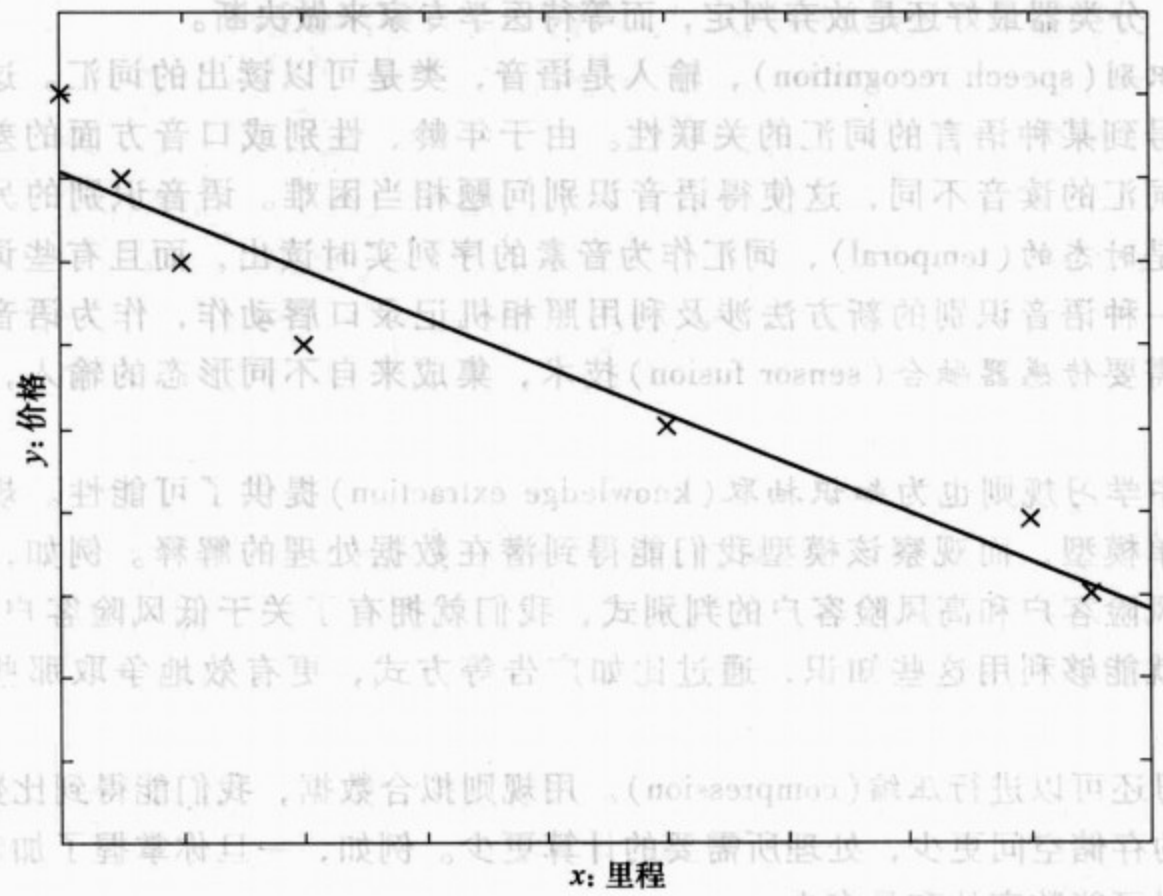


图 1-2 二手车的训练数据及其拟合函数。为简单起见，这里采用线性模型，输入属性也只有里程

回归的另一个例子是对移动机器人的导航。例如，自动汽车导航。其中输出是每次转动车轮的角度，使得汽车前进而不会撞到障碍物或偏离车道。这种情况下，输入由汽车上的传感器(如视频相机、GPS 等)提供。训练数据可以通过监视和记录驾驶员的动作收集。

我们来想象回归的其他应用，这里我们试图优化一个函数[⊖]。假设我们想要造一个焙炒咖啡的机器，该机器有多个影响咖啡品质的输入：各种温度、时间、咖啡豆种类等配置。我们针对不同的输入配置进行大量试验，并测量咖啡的品质，例如，根据消费者的满意度测量咖啡的品质。为寻求最优配置，我们拟合一个联系这些输入和咖啡品质的回归模型，并在当前模型的最优样本附近选择一些新的点，以便寻找更好的配置。我们抽取这些点，检测咖啡的品质，将它们加入训练数据，并拟合新的模型。这通常被称为响应面设计(response surface design)。

9

1.2.4 非监督学习

在监督学习中，我们的目标是学习从输入到输出的映射关系，其中输出的正确值已经由指导者提供。然而，非监督学习中却没有这样的指导者，只有输入数据。我们的目标是发现输入数据中的规律。输入空间存在着某种结构，使得特定的模式比其他模式更常出现，而我们希望知道哪些经常发生，哪些不经常发生。在统计学中，这称为密度估计(density estimation)。

密度估计的一种方法是聚类(clustering)，其目标是发现输入数据的簇或分组。对于拥有老客户数据的公司，客户数据包括客户的个人统计信息，及其以前与公司的交易，公司也许想知道其客户的分布，搞清楚什么类型的客户会频繁出现。这种情况下，聚类模型会将属性

⊖ 感谢 Michael Jordan 提供这个例子。

相似的客户分派到相同的分组，为公司提供其客户的自然分组。一旦找出了这样的分组，公司就可能做出一些决策，比如对不同分组的客户提供特别的服务和产品等。这样的分组也可以用于识别“离群点”，即那些不同于其他客户的客户，这可能意味着一块新的市场，公司可以进一步开发。

聚类的一个有趣的应用是图像压缩(image compression)。在这种情况下，输入实例是由RGB值表示的图像像素。聚类程序将颜色近似的像素点分到相同的分组，而这样的分组对应图像中频繁出现的颜色。如果图像中只有少数颜色，并且我们用一种颜色(例如颜色的平均值)对属于同一分组的像素进行编码，则图像被量化。假设像素是24位，表示1 600万种颜色，但是如果只有64种主色调，那么对于每个像素，我们只需要6位而不是24位。例如，如果景象在图像的不同部分有多种不同的蓝色色调，并且我们采用它们的平均值来表示所有这些蓝色，那么我们就丢失了图像的细节，但是赢得了存储和传送图像的空间。理想状态下，人们希望通过分析重复的图像模式(如纹理、对象等)来识别更高层次的规律性。这为更高层次、更简单、更有用地描述景象提供了可能，并且实现了比像素级更好的压缩。如果我们扫描了文档页，我们得到的不是一些随机的有/无像素，而是一些字符的位图。这样 10
的数据是有结构的，并且我们利用这些冗余信息找出数据的较短描述：“A”的16×16的位图占32字节，其ASCII码只占1个字节。

机器学习方法还应用于生物信息学(bioinformatics)。在我们的基因组中，DNA是“生命的蓝图”，也是碱基即A、G、C和T的序列。RNA由DNA转录而来，蛋白质由RNA转换而来。蛋白质就是生命体和生命体的产物。正如DNA是碱基序列，蛋白质则是氨基酸(由碱基定义)序列。计算机科学在分子生物学的应用领域之一就是比对(alignment)，即将一个序列与另一个序列匹配。这是一个困难的串匹配问题，因为序列可能相当长，有很多模板串要进行匹配，并且还可能会被删节、插入和置换。聚类用于学习结构域(motif)，这是蛋白质结构中反复出现的氨基酸序列。结构域之所以令人感兴趣，是因为它们可能对应它们所表征的序列内部的结构或功能要素。比方说，如果氨基酸是字母，蛋白质是句子，那么结构域就像单词，即具有特别意义、频繁地出现在不同句子中的一串字母。

1.2.5 增强学习

在某些应用中，系统的输出是动作(action)的序列。在这种情况下，单个的动作并不重要，重要的是策略(policy)，即达到目标的正确动作的序列。不存在中间状态中最好动作这种概念。如果一个动作是好的策略的组成部分，那么该动作就是好的。这种情况下，机器学习程序就应当能够评估策略的好坏程度，并从以往好的动作序列中学习，以便能够产生策略。这种学习方法称为增强学习(reinforcement learning)算法。

游戏(game playing)是一个很好的例子。在游戏中，单个移动本身并不重要，正确的移动序列才是重要的。如果一个移动是一个好的游戏策略的一部分，则它就是好的。游戏是人工智能和机器学习的重要研究领域，这是因为游戏容易描述，但又很难玩好。像国际象棋这样的游戏，其规则只有少量的几条，但是它非常复杂，因为在每种状态下都有大量可行的移动，并且每局又都包含有大量的移动。一旦有了能够学习如何玩好游戏的好算法，我们也可 11
以将这些算法用在具有更显著经济效益的领域。

用于在某种环境下搜寻目标位置的机器人导航是增强学习的另一个应用领域。在任何时候，机器人都能够朝着多个方向之一移动。经过多次的试运行，机器人应当学到正确的动作序列，尽可能快地从某一初始状态到达目标状态，并且不会撞到任何障碍物。致使增强学习难度增加的一个因素是系统具有不可靠和不完整的感知信息。例如，装备视频照相机的机器人就得不到完整的信息，因此该机器人总是处于*部分可观测*(partially observable)状态，并且应当将这种不确定性考虑在内。一个任务还可能需要*多智能主体*(multiple agents)的并行操作，这些智能主体将相互作用并协同操作，以便完成一个共同的目标。机器人足球是这种情况的例子之一。

1.3 注释

进化是形成我们的身体形状和我们的内在本能的主要力量。我们还需要终生学习，以改变我们的行为。这有助于我们适应进化论还不能预测的环境变化。在合适的环境下，具有短暂寿命的生物体可能具备它们所有天生的行为能力，但是上苍并未赋予我们应对在有限生命中可能遇见的所有状况的能力。但是，进化赋予我们大脑和学习机制，使得我们可以根据经验实现自我更新，从而适应各种各样的环境。当我们在特定情境下学习到最好的策略时，知识就存储在我们的大脑里。当情境再现，当我们再认知(“认知”意味认出)情境时，我们就能够回忆起合适的策略并采取相应的动作。不过，学习有其局限性；就我们大脑有限的容量来说，也许有些东西我们永远都不可能学会，正像我们永远不可能“学会”长出第三只手臂，或是在脑袋后面长只眼睛，即使它们是有用的我们也学不会。关于心理学视角下的学习和认知可参看 Leahey 和 Harris 1997。注意，与心理学、认知科学以及神经系统科学都不同，机器学习的目标并不是理解人类和动物学习的过程，而是像任何的工程领域一样，机器学习
12 旨在构建一个有用的系统。

几乎所有的科学都在用模型拟合数据。科学家们设计实验、进行观测并收集数据。然后，通过找寻能解释所观测数据的简单模型，尝试抽取知识。该过程称为*归纳*(induction)，它是从一组特别的示例中提取通用规则的过程。

现在，这样的数据分析已经不能再依赖人工完成了，原因有二：一是数据量巨大，二是能够做这种分析的人非常少而且人工分析又很昂贵。因而，对于能够分析数据并自动从中提取信息的计算机模型，也就是说对于学习，人们的兴趣正在不断地增长。

下面章节中我们将要讨论的方法源于不同的科学领域。有时，相同的算法会在多个领域中沿着各自不同的历史轨迹被独立地发现。

在统计学中，从特殊的观测到一般的描述称为*推断*(inference)，而学习称为*估计*(estimation)。分类在统计学中称为*判别式分析*(discriminant analysis)(McLachlan 1992；Hastie、Tibshirani 和 Friedman 2001)。在计算机价格低廉并且数量充足之前，统计学家只能处理小样本。作为数学家，统计学家主要使用能够精确分析的简单参数模型。在工程学中，分类称为*模式识别*(pattern recognition)，方法是非参数的，并且更大程度是凭借经验的(Duda、Hart 和 Stork 2001；Webb 1999)。机器学习与*人工智能*(artificial intelligence)有关(Russell 和 Norvig 1995)，原因是智能系统应当能够适应其环境的变化。视觉、语音以及机器人等应用领域都是从样本数据中学习。在电子工程领域，*信号处理*(signal processing)的研究使得自适应计算

机视觉和语音程序出现。其中，隐马尔可夫模型(Hidden Markov Models，HMM)的发展对于语音识别尤其重要。

20世纪80年代后期，随着VLSI技术的发展和制造含有数千个处理器的并行硬件的可能性的出现，基于多处理单元的分布式计算理论的可行性使得人工神经网络(artificial neural networks)研究领域得到了重生(Bishop，1995)。随着时间的推移，人们认识到在神经网络研究领域中，大多数的神经网络学习算法都具有其统计学的基础(例如多层感知器就是另一类的非参估计)，因此模拟人脑计算的说法开始逐渐淡出。

数据挖掘(data mining)的命名来源于机器学习算法在商界海量数据上的应用(Weiss 和 Indurkhya 1998)。在计算机科学领域，数据挖掘也称为数据库知识发现(knowledge discovery in databases，KDD)。

在统计学、模式识别、神经网络信号处理、控制、人工智能以及数据挖掘等不同领域中，研究工作遵循着各自的途径，并有其各自的侧重点。本书的目标是结合所有这些研究重点，以给出统一的处理问题方法，并提出求解方案。

1.4 相关资源

机器学习的最新研究成果会发表在不同领域的会议和期刊上。机器学习专门的期刊有*Machine Learning*(机器学习)和*Journal of Machine Learning Research*(机器学习研究)。以神经网络为主的期刊有*Neural Computation*(神经计算)、*Neural Networks*(神经网络)以及*IEEE Transactions on Neural Networks*(IEEE神经网络汇刊)。统计学方面的期刊如*Annals of Statistics*(统计学年鉴)和*Journal of the American Statistical Association*(美国统计学会杂志)也会发表一些机器学习方面的文章。另外，*IEEE Transactions on Pattern Analysis and Machine Intelligence*(IEEE模式分析与机器智能汇刊)也是机器学习研究性文章的资源之一。

关于人工智能、模式识别、模糊逻辑以及信号处理方面的期刊也包含机器学习方面的文章。以数据挖掘为主的期刊有*Data Mining and Knowledge Discovery*(数据挖掘与知识发现)、*IEEE Transactions on Knowledge and Data Engineering*(IEEE知识与数据工程汇刊)以及*ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations Journal*(*ACM*知识发现和数据挖掘特别兴趣组期刊)。

关于机器学习方面的主要会议有“*Neural Information Processing Systems*”(NIPS)、“*Uncertainty in Artificial Intelligence*”(UAI)、“*International Conference on Machine Learning*”(ICML)、“*European Conference on Machine Learning*”(ECML)以及“*Computational Learning Theory*”(COLT)、“*International Joint Conference on Artificial Intelligence*”(IJCAI)等。另外，关于神经网络、模式识别、模糊逻辑以及遗传算法等方面的会议，以及关于诸如计算机视觉、语音技术、机器人和数据挖掘等应用方面的会议，也会有针对机器学习的专题。

因特网上有很多数据集，致力于机器学习的研究者经常把它们作为自己研究的基准。下面是一些常用的网址：

- *UCI*机器学习数据库是最流行的数据库：http://www.ics.uci.edu/~mlearn/MLRepository.html
- *UCI KDD Archive*：http://kdd.ics.uci.edu/summary.data.application.html

- *Statlib*: http://lib.stat.cmu.edu
- *Delve*: http://www.cs.utoronto.ca/~delve/

机器学习研究者近期的文章大多数都可以从因特网上找到，http://citeseer.nj.nec.com/cs 上的 NEC 研究索引是一个不错的网站入口。

1.5 习题

1. 设想你有两种选择：可以传真一份文档，即传送图像；或者先使用光学字符阅读器（OCR），然后再传送相应的文本文件。用对比方式论述这两种方法的优缺点，并讨论什么时候一种方法比另一种方法更好。
2. 假定我们正在构建一个 OCR，并且对于每一个字符，我们都存储该字符的位图作为与逐个像素读取的字符匹配的模板。请解释什么时候这样的系统会失败？为什么条码读取器目前仍在使用？
3. 假定我们的既定目标是构建识别垃圾邮件的系统。请问是垃圾邮件中的什么特征使得我们能够确认它为垃圾邮件？计算机如何通过语法分析来发现垃圾邮件？如果发现了垃圾邮件，你希望计算机如何处理它：自动删除？转到另一个文件夹？还是仅仅在屏幕上标亮显示？
4. 如果给定任务是制造自动出租车，请定义约束。输入是什么？输出是什么？如何与乘客沟通？需要与其他的自动出租车沟通，即需要某种语言吗？
5. 在购物篮分析中，我们希望找出产品 X 和 Y 二者之间的依赖关系。对于给定的顾客交易数据库，如何能够发现这些数据之间的依赖关系？进而，如何将依赖关系发现算法推广到多于两个的产品之间？
6. 怎样能够预测用户下一次将键入的命令？或者，怎样能够预测 Web 上下一个将要被下载的网页？这样的预测什么时候是有用的？什么时候会变得令人讨厌？

15

1.6 参考文献

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, 2nd ed. New York: Wiley.

Hand, D. J. 1998. "Consumer Credit and Statistics." In *Statistics in Finance*, ed. D. J. Hand and S. D. Jacka, 69 – 81. London: Arnold.

Hastie, T., R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer.

Leahey, T. H., and R. J. Harris. 1997. *Learning and Cognition*, 4th ed. New York: Prentice Hall.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Russell, S., and P. Norvig. 1995. *Artificial Intelligence: A Modern Approach*. New York: Prentice Hall.

Webb, A. 1999. *Statistical Pattern Recognition*. London: Arnold.

Weiss, S. M., and N. Indurkhya. 1998. *Predictive Data Mining: A Practical Guide*. San Francisco: Morgan Kaufmann.

16

第2章 监督学习

我们从最简单的情况开始讨论监督学习，首先从正例和负例集合中学习类别，继而推广并讨论多类的情况，然后再讨论输出为连续值的回归。

2.1 由实例学习类

假设我们要学习“家用汽车”类 C。现在有一组汽车实例和一组被测人，我们向被测人示以这些汽车。被测人看到所示汽车并标记汽车，将他们认为是家用汽车的标为正例(positive example)，其他的标为负例(negative example)。类学习就是找寻一个涵盖所有的正例而不包括任何负例的描述。通过这些，我们可以做预测：给定一辆我们以前从未见过的汽车，检查学习得到的描述，我们就可以判断这辆汽车是否为家用汽车。我们还可以做知识提取：这种研究可能由汽车公司赞助，目的可以是了解人们对家用汽车的期望。

经过与该领域专家的一些沟通，假定我们得到了一个结论：在我们所掌握的汽车的所有特征中，区别家用汽车与其他汽车的特征是价格和发动机功率。这两个属性就是类识别器的输入(input)。注意，当我们决定采用这种特殊输入表示(input representation)时，我们忽略其他属性，将它们看作是不相关的。尽管有人可能认为诸如座位数量、车身颜色等属性对于辨别车型也很重要，但是这里为了简单起见，我们只考虑价格和发动机功率。

我们假设价格为第一个输入属性 x_1(比如以美元计算)，发动机功率为第二个输入属性 x_2(比如以立方厘米计发动机排量)。这样，每辆汽车就可以用两个数值来表示

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.1}$$

而它的标号表示汽车的类型

$$r = \begin{cases} 1 & \text{如果 } \boldsymbol{x} \text{ 是正例} \\ 0 & \text{如果 } \boldsymbol{x} \text{ 是负例} \end{cases} \tag{2.2}$$

每辆汽车用一个这种有序对$(\boldsymbol{x}, r)$来表示，而训练集中包括 N 个这样的实例

$$\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_{t=1}^{N} \tag{2.3}$$

其中，t 用于标记训练集中的各个汽车实例，它不表示时间或任何类似的序。

现在，我们的训练数据可以绘制在二维空间(x_1, x_2)上，其中每个实例 t 是一个数据点，坐标为(x_1^t, x_2^t)，其类型(即正或负)由 r^t 给定(参见图 2-1)。

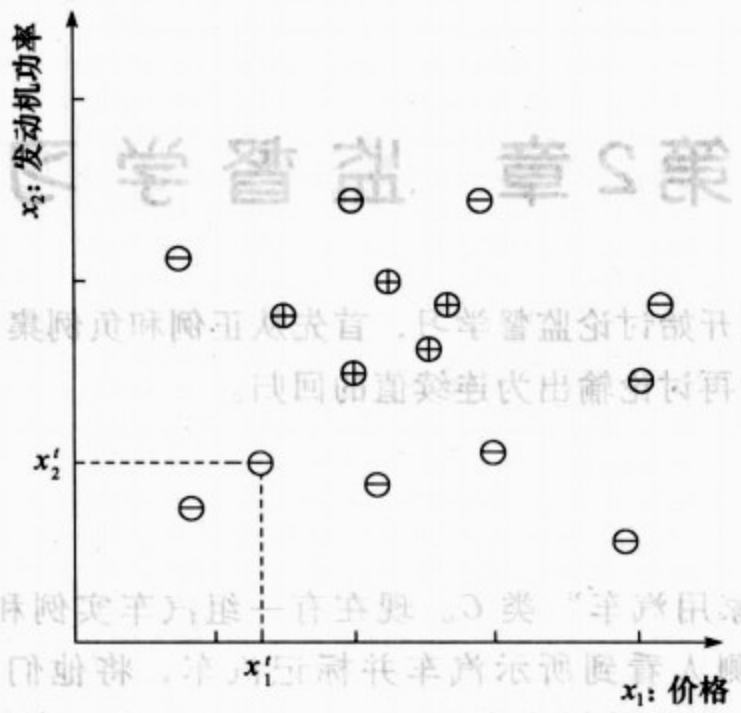


图 2-1 “家用汽车”类的训练集。其中每个点代表一个汽车实例，点的坐标值分别表示汽车的价格和发动机功率。“+”表示正例(家用汽车)，“-”表示负例(非家用汽车)，即其他类型的汽车

通过进一步与专家讨论和分析数据，我们有理由相信，对于家用汽车，其价格和发动机功率应当是在某个确定的范围内

$$(p_1 \leqslant \text{价格} \leqslant p_2)\text{AND}(e_1 \leqslant \text{发动机功率} \leqslant e_2) \tag{2.4}$$

其中 p_1，p_2，e_1 和 e_2 为适当的值。这样，(2.4)式假定类 C 是价格-发动机功率空间中的矩形(参见图 2-2)。

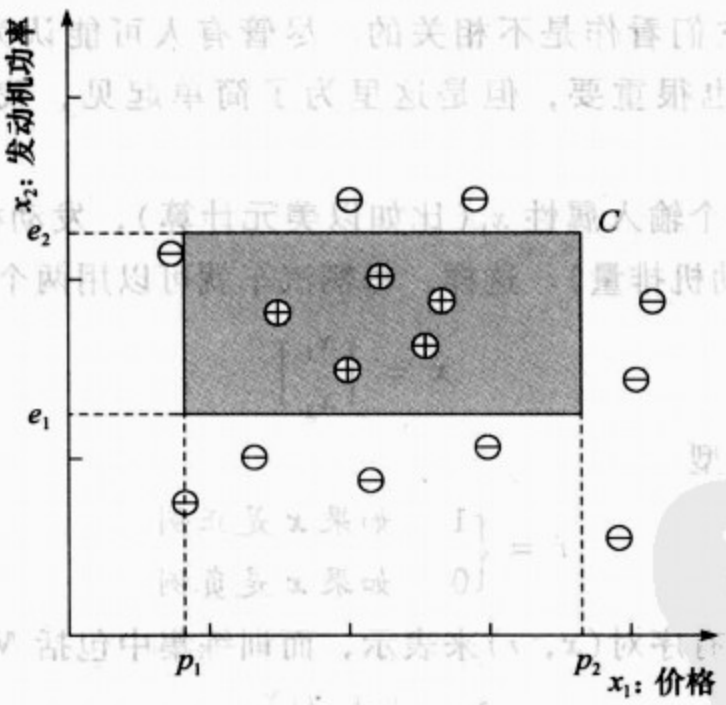


图 2-2 假设类的实例。家用汽车类是价格和发动机功率空间中的矩形

(2.4)式确定了假设类(hypothesis class)$\mathcal{H}$(即矩形的集合)，我们相信 C 是从中抽取的。学习算法应当找到一个特定的假设(hypothesis)$h \in \mathcal{H}$，尽可能地逼近 C。

18 ~ 19

尽管专家定义了假设类，但是他却不能说出参数值是什么。换句话说，尽管我们选定了 $\mathcal{H}$，但是我们却不知道哪个特定的 $h \in \mathcal{H}$ 等于或最接近 C。然而，一旦我们把注意力局限于

这个假设类，学习类就归结为较简单的问题——找出定义 h 的 4 个参数。

我们的目标是找出 $h \in \mathcal{H}$，它与 C 尽可能类似。假设 h 对实例 $\boldsymbol{x}$ 进行预测，使得

$$h(\boldsymbol{x}) = \begin{cases} 1 & \text{如果 } h \text{ 将 } \boldsymbol{x} \text{ 分类为正例} \\ 0 & \text{如果 } h \text{ 将 } \boldsymbol{x} \text{ 分类为负例} \end{cases} \tag{2.5}$$

实际上我们并不知道 $C(\boldsymbol{x})$，因此也无法评估 $h(\boldsymbol{x})$ 与 $C(\boldsymbol{x})$ 的匹配程度。我们所拥有的是训练集 $\mathcal{X}$，它是所有可能的 $\boldsymbol{x}$ 的一个小子集。经验误差（empirical error）是 h 的预测值（prediction）与 $\mathcal{X}$ 中给定的预期值（required value）不同的训练实例所占的比例。对于给定的训练集 $\mathcal{X}$，假设 h 的误差是

$$E(h \mid \mathcal{X}) = \sum_{t=1}^{N} 1(h(x^t) \neq r^t) \tag{2.6}$$

其中，当 $a \neq b$ 时 $1(a \neq b)$ 为 1，当 $a = b$ 时 $1(a \neq b)$ 为 0（参见图 2-3）。

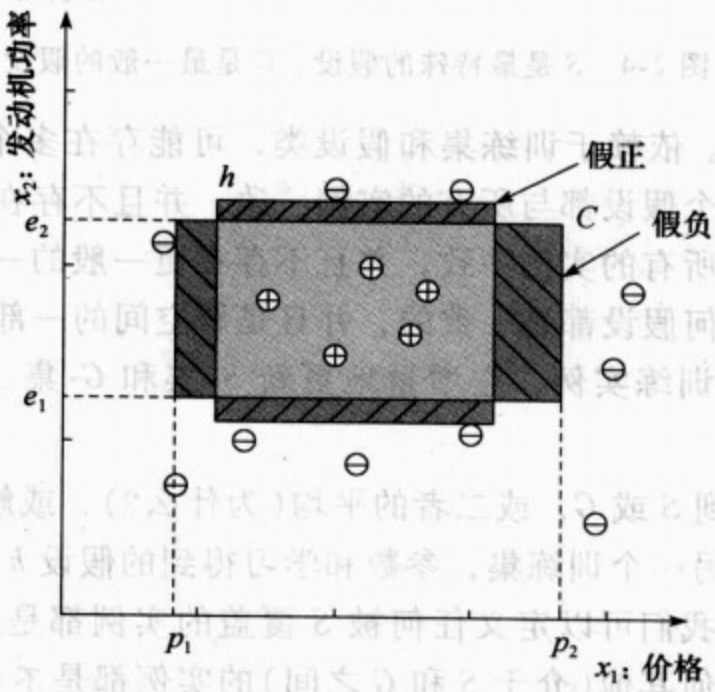


图 2-3 C 是实际的类，h 是我们的诱导假设。C 为 1 而 h 为 0 的点为假负，C 为 0 而 h 为 1 的点为假正。其他点，即真正和真负，都被正确地分类

在我们的例子中，假设类 $\mathcal{H}$ 是所有可能的矩形的集合。每个四元组 $(p_1^h, p_2^h, e_1^h, e_2^h)$ 都定义 $\mathcal{H}$ 中的一个假设 h，而我们需要选择其中最好的一个；换句话说，给定训练集，我们需要找出这四个参数的值，使得它涵盖所有的正例而不包括任何的负例。注意，如果 x_1 和 x_2 是实数，则存在无穷多个 h 满足上述条件，也就是说，对于这些 h 误差 E 为零。但是，给定一个接近于正例和负例边界的某个未来实例，不同的候选假设可能做出不同的预测。这是泛化（generalization）问题，即我们的假设对不在训练集中的未来实例的分类的准确率如何。

一种可能的策略是找出最特殊的假设（most specific hypothesis）S，它是涵盖所有正例而不包括任何负例的最紧凑的矩形（参见图 2-4）。这样就得出一个假设 $h = S$，作为我们的诱导类（induced class）。注意实际的类 C 可能会比 S 更大但绝不会更小。最一般的假设（most general hypothesis）G 是涵盖所有正例而不包括任何负例的最大矩形（参见图 2-4）。对于任何介于 S 和 G 之间的 $h \in \mathcal{H}$，h 为无误差的有效假设，称作与训练集相容（consistent），并且这样的 h 形成解空间（version space）。

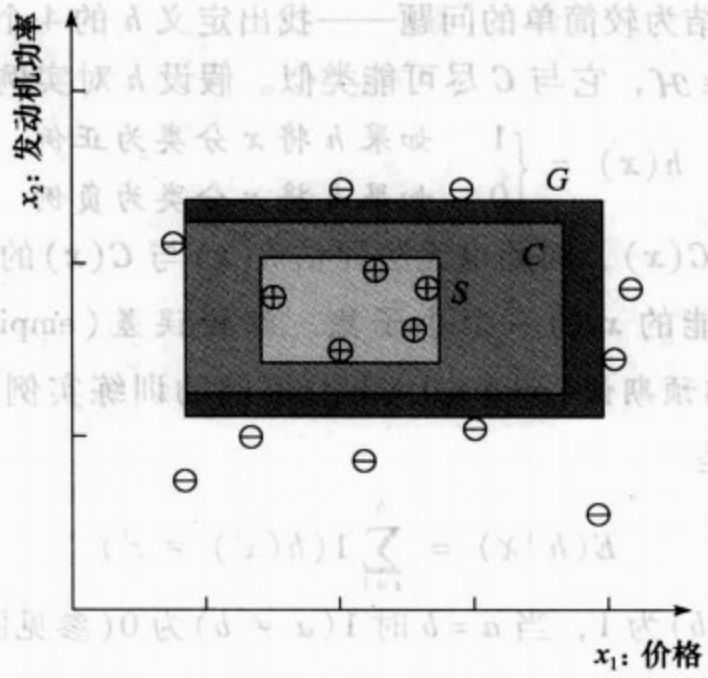


图 2-4 S 是最特殊的假设，G 是最一般的假设

S 和 G 不必是唯一的。依赖于训练集和假设类，可能存在多个 S_i 和 G_i，它们分别形成 S-集和G-集。S-集中的每个假设都与所有的实例一致，并且不存在更特殊的一致假设。类似地，G-集中的每假设都与所有的实例一致，并且不存在更一般的一致假设。这两个集合形成边界集，而它们之间的任何假设都是一致的，并且是解空间的一部分。存在一个称作候选删除的算法，随着逐个看到训练实例，它增量地更新 S-集和 G-集。见 Mitchell 1997，Russell 和 Norvig 1995。[⊖]

给定$\mathcal{X}$，我们可以找到 S 或 G，或二者的平均(为什么?)，或解空间中的任意 h，并将它作为我们的假设 h。给定另一个训练集，参数和学习得到的假设 h 可能不同。

作为另外一种可能，我们可以定义任何被 S 覆盖的实例都是正例，而任何没被 G 覆盖的实例都是负例，并且任何其他(介于 S 和 G 之间)的实例都是不确定(doubt)实例，由于缺乏数据支持，无法正确标注这些不确定实例。在这种情况下，系统将去除 (reject) 这些实例，并留待人类专家来判定。

这里，我们假定$\mathcal{H}$包含 C，即存在 $h \in \mathcal{H}$，使得 $E(h|\mathcal{X})$为 0。给定假设类$\mathcal{H}$，可能存在不能学习 C 的情况，即不存在 $h \in \mathcal{H}$，使得误差为 0。因此，对于任何的应用，我们都需要确信$\mathcal{H}$有足够的柔性，或$\mathcal{H}$具有足够的“能力”学习 C。

2.2 VC 维

假定我们有一个数据集，包含 N 个点。这 N 个点可以用 2^N 种方法标记为正例和负例。因此，N 个数据点可以定义 2^N 种不同的学习问题。如果对于这些问题中的任何一个，我们都能够找到一个假设 $h \in \mathcal{H}$将正例和负例分开，那么我们就称$\mathcal{H}$散列(shatter)N 个点。也就是说，可以用 N 个点定义的任何的学习问题都能够用一个从$\mathcal{H}$中抽取的假设无误差地学习。可以被$\mathcal{H}$散列的点的最大数量称为$\mathcal{H}$的 VC 维(Vapnik-Chervonenkis dimension)，记为

⊖ 这一段文字是作者勘误表提供的对 S 和 G 的进一步解释，根据作者的意见插在此处。——译者注

VC($\mathcal{H}$)，它度量假设类$\mathcal{H}$的学习能力（capacity）。

在图 2-5 中，我们可以看到，轴平行的矩形能够散列二维空间的 4 个点。因此，当$\mathcal{H}$为二维空间中轴平行的矩形的假设类时，VC($\mathcal{H}$)等于 4。在计算 VC 维时，能找到 4 个被散列的点就够了；没有必要去散列二维空间中任意 4 个点。例如，位于同一直线上的 4 个点不能被矩形散列。然而，我们无法在二维空间的任何位置设置 5 个点，使得对于所有可能的标记，一个矩形能够分开正例和负例。

20 ~ 22

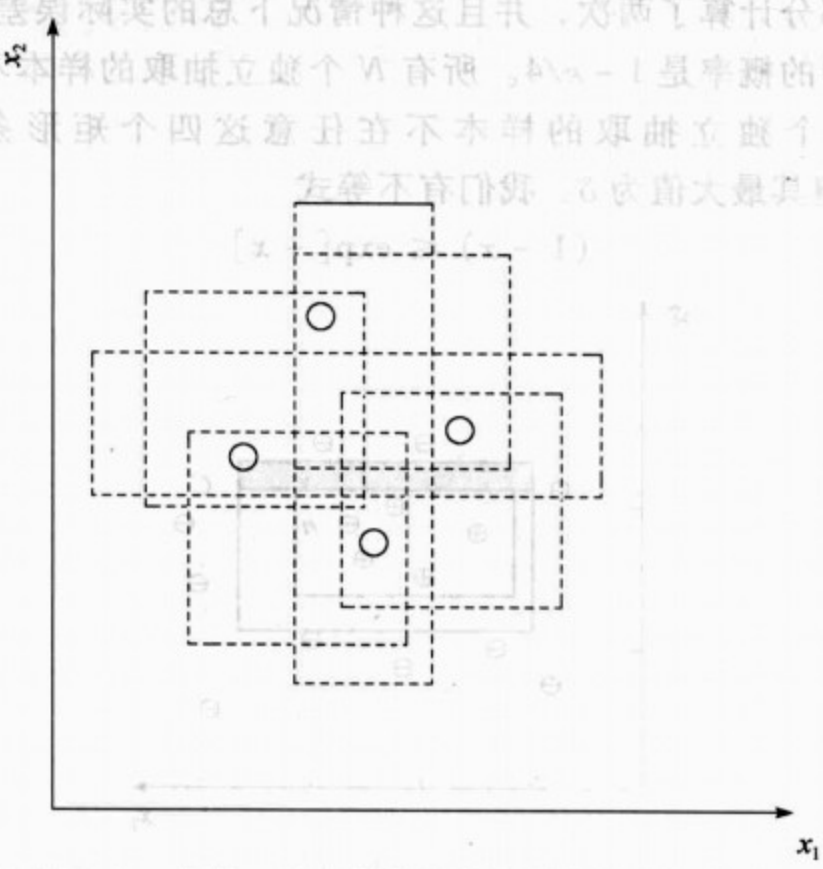


图 2-5　轴平行的矩形能够散列 4 个点，其中只显示了覆盖两个点的矩形

也许 VC 维看起来比较悲观，它告诉我们使用矩形作为假设类，我们只能学习包括 4 个点的数据集。能够学习含有 4 个点的数据集的学习算法不是很有用。然而，这是因为 VC 维独立于数据实例的概率分布。在实际生活中，世界是平滑变化的，在大多数时间相近的实例具有相同的标记，我们并不需要担心所有可能的标记。有很多包含远不止 4 个点的数据集都可以通过我们的假设类来学习(参见图 2-1)。因此，即便是具有较小 VC 维的假设类也是有应用价值的，并且比那些较大的 VC 维(例如，具有无穷 VC 维的查找表)更可取。

23

2.3　概率逼近正确学习

使用最紧凑的矩形 S 作为假设，希望找出我们需要多少实例。我们希望我们的假设是近似正确的，即误差概率不超过某个值。还要对我们的假设有信心，因为我们想知道我们的假设在大多数时间里都是正确的。因此我们希望我们的假设(以我们可以指定的概率)是正确的。

在*概率逼近正确*(Probably Approximately Correct，PAC)学习中，给定类 C 和从未知但具有确定概率分布 $p(x)$ 中抽取的样本，我们希望找出样本数 N，使得对于任意的 $\delta \leqslant 1/2$ 和

$\varepsilon > 0$，假设 h 的误差至多为 ε 的概率至少为 $1-\delta$。

$$P\{C\Delta h \leqslant \varepsilon\} \geqslant 1-\delta$$

其中，$C\Delta h$ 是 C 与 h 不同的区域。

在这种情况下，因为 S 是最紧凑的可能的矩形，C 与 $h=S$ 之间的误差区域是四个矩形条带之和(参见图2-6)。我们希望确保正例落在该区域(导致错误)的概率最多为 ε。对于任何这样的条带，如果我们能够确保其概率上界为 $\varepsilon/4$，则误差最多为 $4(\varepsilon/4)=\varepsilon$。注意，我们将矩形角部的重叠部分计算了两次，并且这种情况下总的实际误差小于 $4(\varepsilon/4)$。随机抽取的样本不在此条带中的概率是 $1-\varepsilon/4$。所有 N 个独立抽取的样本不在此条带中的概率为 $(1-\varepsilon/4)^N$，所有 N 个独立抽取的样本不在任意这四个矩形条带中的概率最多为 $4(1-\varepsilon/4)^N$，我们希望其最大值为 δ。我们有不等式

$$(1-x) \leqslant \exp[-x]$$

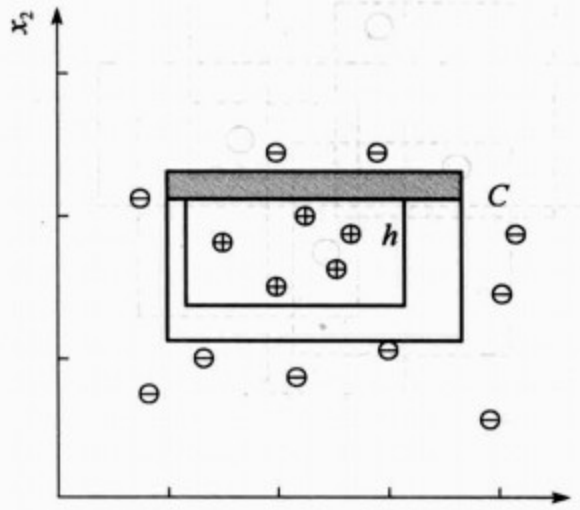


图2-6 h 与 C 之差是四个矩形条带之和，其中一个用阴影显示

因此，如果选定 N 和 δ 满足

$$4\exp[-\varepsilon N/4] \leqslant \delta$$

则我们有 $4(1-\varepsilon/4)^N \leqslant \delta$。不等式两边同时除以4，再取(自然)对数，并重新排列各项，我们得到

$$N \geqslant (4/\varepsilon)\log(4/\delta) \tag{2.7}$$

因此，只要我们至少从 C 中取 $(4/\varepsilon)\log(4/\delta)$ 个独立样本，并使用紧凑矩形作为我们的假设 h，则在置信概率(confidence probability) 至少为 $1-\delta$ 的情况下，一个给定点被误分类的错误概率最多为 ε。减少 δ 我们可以有任意大的置信度，而减少 ε 我们可以有任意小的误差，并且我们在不等式(2.7) 中看到，样本的数量是分别随 $1/\varepsilon$ 和 $1/\delta$ 呈线性和对数缓慢增长的函数。

2.4 噪声

噪声(noise)是数据中有害的异常。由于噪声的存在，类的学习可能更加困难，并且使用简单的假设可能做不到零误差(参见图2-7)。噪声有以下几种解释：

- 记录输入属性可能不准确，这可能导致数据点在输入空间的移动。
- 标记数据点可能有错，可能将正例标记为负的，或相反。这种情况称为指导噪声(teacher noise)。
- 可能存在我们没有考虑到的附加属性，而它们会影响实例的标注。这些附加属性可能是隐藏的(hidden)或潜在的(latent)，因此是不可观测的。这些被忽略的属性所造成的影响作为随机成分，是“噪声”的一部分。

如图 2-7 所示，当有噪声时，在正负实例之间不存在简单的边界，并且为了将它们分开，我们需要复杂的假设以对应能力更大的假设类。矩形可以用 4 个数定义，然而为了定义更复杂的形状，我们就需要具有大量参数的更复杂的模型。利用这些复杂模型，我们可以更好地拟合数据，得到零误差(参见图 2-7 中的曲线图形)。另一个可行的方法是保持模型的简单性并允许一些误差的存在(参见图 2-7 中的矩形)。

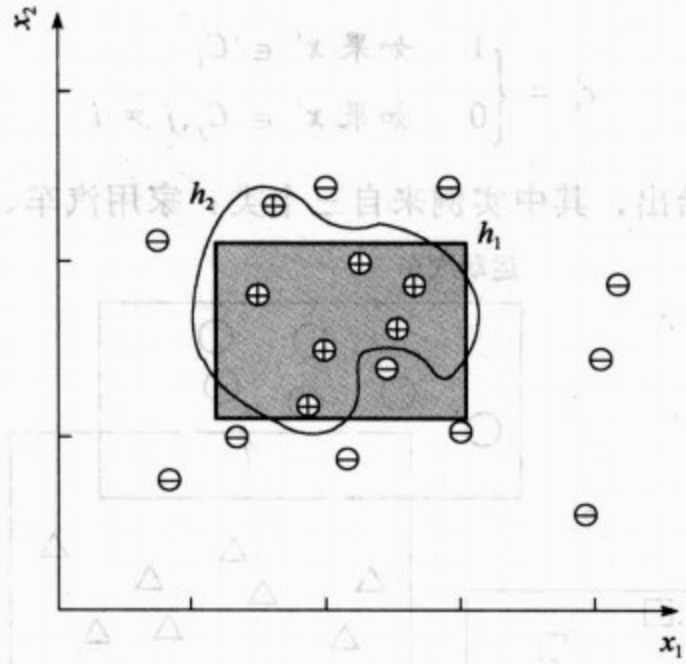


图 2-7 当有噪声时，在正例和负例之间不存在一个简单的边界，使用简单假设也许不可能达到零误差的分类结果。矩形是具有 4 个定义隅角的参数的简单假设。使用大量控制点的分段函数能够导出任意的封闭图形

使用简单的矩形(除非其训练误差很大)更有意义，原因如下：

1. 矩形是一种容易使用的简单模型。容易检查一个点是在矩形内还是在矩形外，并且对于未来的数据实例，我们都可以容易地检查它是正例还是负例。

2. 矩形是一种容易训练的简单的模型，并且具有较少的参数。相对任意图形的控制点来说，比较容易找到矩形的隅角值。利用小规模训练集，当训练实例有少许差异时，我们预期简单模型比复杂模型变化小一些：简单模型具有更小的*方差*(variance)。另一方面，太简单的模型假设更多、更严格，并且如果潜在类并非那么简单，模型预测就可能失败：较简单的模型具有较大的偏倚(bias)。求解最优模型相当于最小化偏倚和方差。

3. 矩形是容易解释的简单模型。矩形简单地对应在两个属性上定义的区间。通过学习简单的模型，我们能够从给定训练集的原始数据中提取信息。

4. 如果输入数据中确实存在错误标记的实例或噪声，并且实际的类确实就是像矩形这样的简单模型，那么由于矩形具有较小的方差，并且较少地被单个实例所影响，所以

尽管简单矩形可能导致训练集上较大的误差，它也是比曲线图形更好的分类器。我们说简单(但不是太简单的)模型会比复杂模型泛化能力更好。该规则就是著名的奥克姆剃刀规则(Occam's razor)，它是说较简单的解释看上去更可信，并且任何不必要的复杂性都应该被摒弃。

2.5 学习多类

26
~
27

在前面的家用汽车例子中，我们有属于家用汽车类的正例和属于其他所有汽车类别的负例。这是一个两类(two-class)问题。通常情况下，我们有 K 个类，记为 C_i，$i=1, \cdots, K$，并且每个输入实例严格地属于其中一个类。训练集形如

$$\mathcal{X} = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_{t=1}^{N}$$

其中 $\boldsymbol{r}$ 是 K 维的，并且

$$r_i^t = \begin{cases} 1 & \text{如果 } \boldsymbol{x}^t \in C_i \\ 0 & \text{如果 } \boldsymbol{x}^t \in C_j, j \neq i \end{cases} \tag{2.8}$$

一个例子在图 2-8 中给出，其中实例来自三个类：家用汽车、运动汽车和豪华轿车。

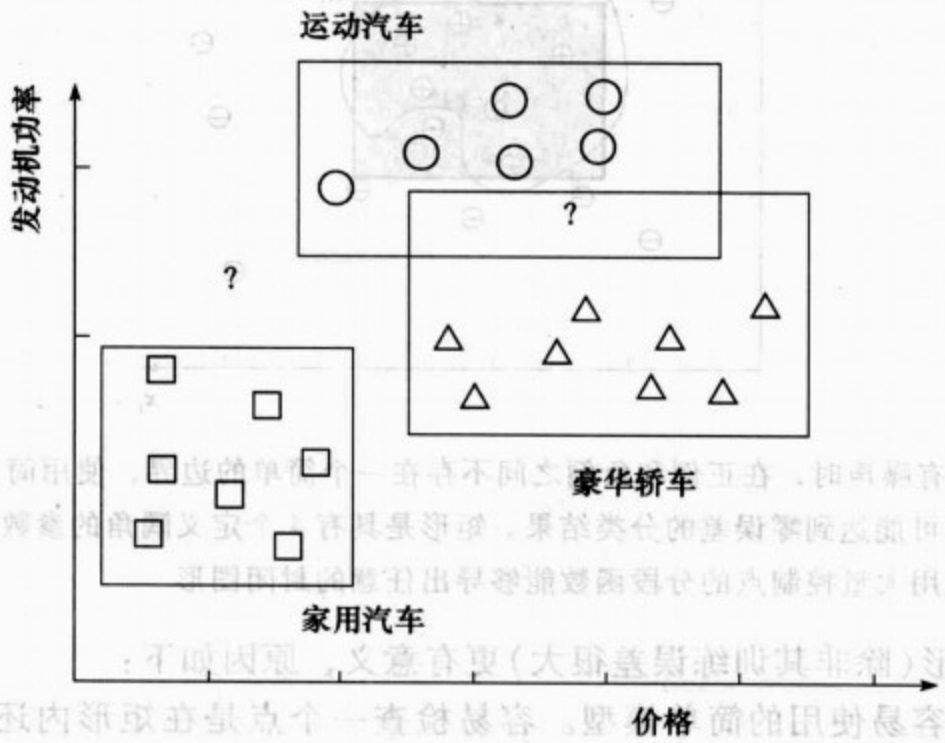


图 2-8 有三个类：家用汽车、运动汽车和豪华轿车。有三个归纳的假设，每个假设覆盖一个类的实例而不包括另外两个类的实例。“?”为拒绝区域，其中没有类或有多个类被选中

在用于分类的机器学习中，我们希望学习将一个类与所有其他类分开的边界。这样，我们把 K-类的分类问题看作是 K 个两类问题。属于 C_i 类的训练实例是假设 h_i 的正例，属于所有其他类的训练实例是假设 h_i 的负例。因此，在 K-类的分类问题中，我们要学习 K 个假设，使得

28

$$h_i(\boldsymbol{x}^t) = \begin{cases} 1 & \text{如果 } \boldsymbol{x}^t \in C_i \\ 0 & \text{如果 } \boldsymbol{x}^t \in C_j, j \neq i \end{cases} \tag{2.9}$$

在理想情况下，对于给定的 $\boldsymbol{x}$，只有其中一个假设 $h_i(\boldsymbol{x})(i=1, \cdots, K)$ 为 1，并且我们能够选定一个类。但是，当没有或者有两个或更多的 $h_i(\boldsymbol{x})$ 为 1 时，我们就无法选定一个

类，这是不确定的(doubt)情况并且分类器要拒绝这种情况。

在学习家用汽车的例子中，我们只用了一个假设，并且只对正例样本建模。任何未包括在其中的实例都不是家用汽车。作为另一种选择，有时我们可能更倾向于构建两个假设，一个是对正例，另一个是对负例。这也为被另一个假设所覆盖的负例假定一个结构。将家用汽车与运动汽车分开就是一个这样的问题，每个类都有其自己的结构。这种处理的优点在于，如果输入的是一辆豪华轿车，我们就能够通过两个假设来判定其为负例并丢弃该输入。

2.6 回归

在分类问题中，给定一个输入，所产生的输出是一个布尔值，这是一个是/否型答案。当输出是数值型的值时，我们所希望学习的不是一个类 $C(\boldsymbol{x}) \in \{0, 1\}$，而是一个连续函数。在机器学习中，函数是未知的，不过我们有从其中抽取的训练集

$$\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_{t=1}^N$$

其中 $r^t \in \Re$。如果不存在噪声，则该任务是插值(interpolation)。我们希望找到通过这些点的函数 $f(\boldsymbol{x})$，使得

$$r^t = f(\boldsymbol{x}^t)$$

在多项式插值（polynomial interpolation）中，给定 N 个点，我们找出可以用来预测任何 $\boldsymbol{x}$ 的输出的 $(N-1)$ 阶多项式。如果 $\boldsymbol{x}$ 落在训练集中 $\boldsymbol{x}^t$ 的值域之外，则该方法称为外插或外推(extrapolation)。例如，在时间序列预测中，我们拥有截至到目前的数据，而且希望预测未来的值。在回归(regression)分析中，噪声添加到未知函数的输出上

$$r^t = f(\boldsymbol{x}^t) + \varepsilon \tag{2.10}$$

29

其中 $f(\boldsymbol{x}) \in \Re$ 是未知函数，ε 是随机噪声。关于噪声的解释是，存在我们无法观察到的额外的隐藏(hidden)变量

$$r^t = f^*(\boldsymbol{x}^t, \boldsymbol{z}^t) \tag{2.11}$$

其中 $\boldsymbol{z}^t$ 表示这些隐藏变量。我们希望通过模型 $g(\boldsymbol{x})$ 来逼近输出。训练集 $\mathcal{X}$ 上的经验误差是

$$E(g \mid \mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}[r^t - g(\boldsymbol{x}^t)]^2 \tag{2.12}$$

因为 r 和 $g(\boldsymbol{x})$ 是数值量(例如，属于 $\Re$)，因此存在定义在其值域上的序，并且我们可以定义值之间的距离(distance)为差的平方。相对于分类使用的等于或不等于来说，距离给我们提供了更多的信息。差的平方是一种可以使用的误差函数，另一种误差函数是差的绝对值。在后续章节中，我们将会看到一些其他例子。

我们的目标是找到最小化经验误差的 $g(\cdot)$。我们的方法又是相同的，我们对 $g(\cdot)$ 假定一个的具有少量参数的假设类。如果假定 $g(\boldsymbol{x})$ 是线性的，则我们有

$$g(\boldsymbol{x}) = w_1 x_1 + \cdots + w_d x_d + w_0 = \sum_{j=1}^{d} w_j x_j + w_0 \tag{2.13}$$

现在，再回到 1.2.3 节的例子，在那里我们估计一辆二手车的价格。当时我们使用单个输入的线性模型

$$g(x) = w_1 x + w_0 \tag{2.14}$$

其中，w_1 和 w_0 是需要从数据中学习的参数。w_1 和 w_0 的值应该使下式最小化

30

$$E(w_1, w_0 \mid \mathcal{X}) = \frac{1}{N}\sum_{t=1}^{N}[r^t - (w_1 x^t + w_0)]^2 \tag{2.15}$$

其最小点可以通过求 E 关于 w_1 和 w_0 的偏导数，令偏导数为0，并求解这两个未知量来计算：

$$w_1 = \frac{\sum_t x^t r^t - \bar{x}\bar{r}N}{\sum_t (x^t)^2 - N\bar{x}^2} \tag{2.16}$$

$$w_0 = \bar{r} - w_1 \bar{x}$$

其中，$\bar{x} = \sum_t x^t/N$，$\bar{r} = \sum_t r^t/N$。找到的直线如图 1-2 所示。

如果线性模型过于简单，它就会太受限制，导致大的近似误差，并且在这种情况下，输出可以取输入的较高阶的函数，例如二次函数

$$g(x) = w_2 x^2 + w_1 x + w_0 \tag{2.17}$$

其中类似地，我们有参数的解析解。当多项式的阶增加时，训练数据上的误差将会降低。但是高阶多项式关注个体样本，而不是捕获数据一般趋势(参看图 2-9 中的六次多项式)。因此，当精确调整的模型复杂性达到潜在数据的函数的复杂度时，我们应该谨慎行事。

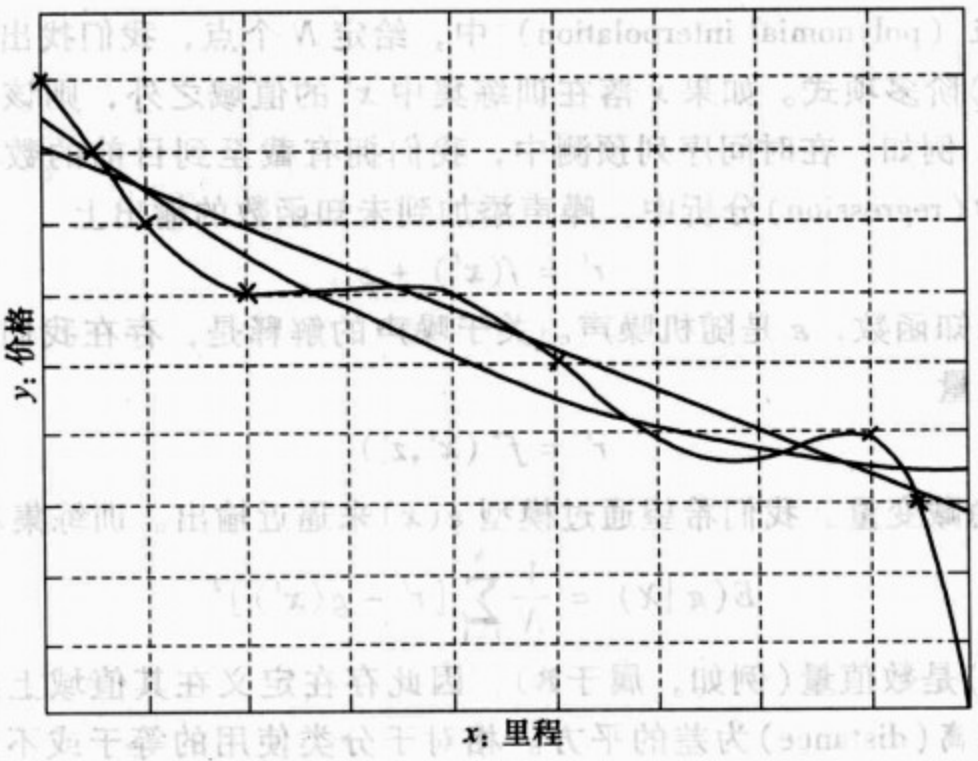


图 2-9 拟合相同的数据点集的线性、二次和六次多项式。最高阶的多项式(六次多项式)给出了正确的拟合，但是给定更多数据，真实的曲线很可能不是这种形状。二次多项式看起来比线性拟合好，它捕获了训练数据的走势

2.7 模型选择与泛化

我们用从实例学习布尔函数作为例子开始。在布尔函数中，所有的输入和输出均为二元的。d 个二元值有 2^d 种可能的写法。因此，对于 d 个输入，训练集最多能有 2^d 个样本实例。如表 2-1 所示，其中的每一位都能标记为 0 或 1，因而对于 d 个输入，将有 2^{2^d} 个可能的布尔函数。

表 2-1 2 个输入存在 4 种可能的情况和 16 种可能的布尔函数

x_1	x_2	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8	h_9	h_{10}	h_{11}	h_{12}	h_{13}	h_{14}	h_{15}	h_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

每一个不同的训练样本都会去掉一半的假设，即去掉那些猜测出错的假设。例如，假定我们有 $x_1=0$，$x_2=1$，而输出为 0，这种情况就去掉了假设 h_5、h_6、h_7、h_8、h_{13}、h_{14}、h_{15}和 h_{16}。这是观察学习的一种途径；随着我们看到更多的训练样例，我们逐步去掉那些与训练数据不一致的假设。在布尔函数的情况下，为了最终得到单个假设，我们需要看到所有的 2^d 个训练样本。如果给定的训练集只包含所有可能实例的一个小子集(通常情况就是如此)，也就是说，如果我们仅对少量情况知道输出应该是什么，则解是不唯一的。看到 N 个样本实例后，还有 2^{2^d-N}个可能的函数。这是一个不适定问题(ill-posed problem)，其中仅靠数据本身不足以找到唯一解。

在其他的学习应用中，在分类、回归中也存在同样的问题。随着我们看到更多的训练实例，我们对潜在函数的了解就更多，并且我们从假设类去掉更多不一致的假设，但是我们还剩下许多一致的假设。

这样，由于学习是一个不适定问题，并且单靠数据本身不足以找到解，因此我们应该做一些特别的假设，以便得到已有数据的唯一解。我们把为了使学习成为可能所做的假设集称为学习算法的归纳偏倚(inductive bias)。引入归纳偏倚的一种途径是假定一个假设类。在学习家用汽车类时，存在着无限种将正例与负例分开的方法。假定矩形是一种归纳偏倚，而后假定最紧凑的矩形就是另外一种归纳偏倚。在线性回归中，假定线性函数也是一种归纳偏倚。

然而，我们知道，每个假设类都有一定的能力，并且只能够学习确定的函数。使用具有更大能力、包含更复杂的假设的假设类，可以扩充可学习的函数类。例如，假设类“两个互不重叠的矩形的并”具有更大的能力，但是其假设也更加复杂。类似地，在回归分析中，随着多项式的阶增高，其能力和复杂性也不断增加。现在的问题是要决定在哪里停止。

因此，如果没有归纳偏倚，则学习将是不可能的，而且现在的问题是如何选择正确的偏倚。该问题称作模型选择(model selection)。对于这种问题的解答，我们应当记住机器学习的目标很少是去复制训练数据，而是预测新情况。也就是说，我们希望对于训练集之外的输入(其正确的输出并没有在训练集中给出)能够产生正确的输出。训练集上训练的模型在多大程度上能够对新的实例预测出正确输出称为泛化(generalization)。

对最好的泛化来说，我们应当使假设的复杂性与潜在数据的函数的复杂性相匹配。如果假设没有函数复杂，例如，当试图用直线拟合从三次多项式抽取的数据时，则是欠拟合(underfitting)。这种情况下，随着复杂性的增高，训练误差和确认误差都会降低。但是，如果我们的假设太过复杂，数据不足以约束该假设，我们最后也许得到不好的假设。例如，当

用两个矩形拟合从一个矩形抽取的数据时，这种情况就会发生。或者如果存在噪声，则过分复杂的假设可能不仅学习潜在的函数，而且也学习数据中的噪声，导致很差的拟合。例如，用六次多项式拟合从三次多项式抽样的噪声数据时，这种情况就会发生。这称为过(分)拟合(overfitting)。在这种情况下，拥有更多的训练数据是有帮助的，但是只能达到一定程度。

31
~
33

我们可以引用三元权衡(triple trade-off)(Dietterich 2003)来总结我们的讨论。在所有的由实例数据训练的学习算法中，存在以下三种因素之间的平衡：

- 拟合数据的假设的复杂性，即假设类的能力；
- 训练数据的总量；
- 在新的实例上的泛化误差。

随着训练数据量的增加，泛化误差降低。随着模型复杂性的增加，泛化误差先降低，然后开始增加。过于复杂的假设的泛化误差可以通过增加训练数据的总量来控制，但是只能达到一定程度。

如果我们访问训练集以外的数据，我们就能够度量假设的泛化能力，即归纳偏倚的质量。我们通过将训练集划分为两个部分来模拟这一过程。我们使用一部分作训练(即找出假设)，剩下的部分称作确认集(validation set)，并用来检验假设的泛化能力。假定训练集和确认集都足够大，则在确认集上最精确的假设就是最好的假设(即具有最佳归纳偏倚的假设)。该过程称为交叉确认(cross-validation)。例如，为了找出多项式回归的正确的阶，给定多个不同阶的候选多项式，我们在训练集上求出它们的系数，在确认集上计算它们的误差，并取具有最小确认误差的多项式作为最佳多项式。

注意，如果需要报告反映我们的最佳模型的期望误差的话，我们就不应该使用确认误差。我们已经使用确认集来选择最佳模型，并且它实际上已经成为训练集的一部分。我们需要第三个数据集——检验集(test set)，有时也称为发布集(publication set)，它包含在训练或确认阶段未使用过的数据。现实生活也有类似的情况，例如我们选修一门课程：老师在讲授一门课时，课堂上求解的例题构成了训练集，考试题目就是确认集，而我们在职业生涯中解决的问题则是检验集。

在第14章，我们将继续讨论如何评估模型的错误率，以及当我们没有大数据集来划分
34 成两部分或三部分时，我们如何从两个模型中选择较好的模型。

2.8 监督机器学习算法的维

现在，让我们来总结并归纳上述要点。我们有样本

$$\mathcal{X} = \{\boldsymbol{x}^t, r^t\}_{t=1}^N \tag{2.18}$$

该样本是独立同分布的(independent and identically distributed, iid)；次序并不重要，而所有的实例都取自相同的联合分布 $p(\boldsymbol{x}, r)$。t 指示 N 个实例中的一个，$\boldsymbol{x}^t$ 是任意维的输入，而 r^t 是相关联的预期输出。对于两类学习，r^t 是0/1；对于 $K(K>2)$ 类问题，r^t 是一个 K 维二元向量(其中恰有一维为1，其他各维均为0)；在回归分析中，r^t 是一个实数值。

我们的目标是使用模型 $g(\boldsymbol{x}^t \mid \theta)$ 来构建一个 r^t 的好的、有用的近似。为了达到预期目标，我们必须做出三个决定：

1. 学习所使用的模型(model)，记作

$$g(x \mid \theta)$$

其中，$g(\cdot)$是模型，x是输入，θ是参数。$g(\cdot)$定义假设类，而θ的特殊值示例了假设类中的一个假设。例如，在类的学习中，我们把矩形当作模型，其4个坐标值构成了θ。在线性回归中，模型是输入的线性函数，其斜率和截距是从数据中学习的参数。模型(归纳偏倚)由机器学习系统的设计者根据其应用知识背景决定，参数由学习算法，利用取样于实际应用的训练集而进行调整。

2. 损失函数(loss function)$L(\cdot)$，用于计算预期输出r^t与给定参数θ的当前值时我们对它的近似$g(x^t \mid \theta)$之间的差。逼近误差(approximation error)或损失(loss)是各单个实例之上的损失之和

$$E(\theta \mid \mathcal{X}) = \sum_t L(r^t, g(x^t \mid \theta)) \tag{2.19}$$

在输出为0/1类的学习中，$L(\cdot)$检测相等或不等；在回归分析中，由于输出是数值型值，我们有关于距离的序信息，而且一种可能性是使用差的平方。 35

3. 最优化过程(optimization procedure)求解最小化近似误差的θ^*

$$\theta^* = \arg\min_\theta E(\theta \mid \mathcal{X}) \tag{2.20}$$

其中arg min返回使E最小化的参数值。在回归中，我们能够解析地求解最优化问题。使用更复杂的模型和误差函数，我们需要使用更加复杂的优化方法。例如，基于梯度的方法、模拟退火或遗传算法等。

为了做好上述工作，必须满足以下条件：首先，假设类$g(\cdot)$应当足够大，即要有足够的容量，以便包含在含噪声情况下产生r^t表示的数据的未知函数。第二，必须有足够的训练数据，使得我们从假设类中识别正确(或足够好)的假设。第三，给定训练数据，我们应当有好的优化方法，以便找出正确的假设。

不同的机器学习方法之间的区别或者在于它们假设的模型(假设类/归纳偏倚)不同，或者在于它们所使用的损失度量不同，或者在于它们所使用的最优化过程不同。我们将在后续的章节中看到更多的例子。

2.9 注释

Mitchell提出了解空间和候选排除算法，使得当样本实例依次给出时，可以增量地构建S和G，近期的评述可参见Mitchell 1997。Hirsh(1990)讨论了当实例样本受到少量噪声影响时，如何处理解空间。有关机器学习最早的研究工作之一是Winston(1975)提出的“几乎错过”(near miss)思想。几乎错过是一个与正例非常相似的负例。用我们的术语，几乎错过就是可能落在S与G之间灰色区域的实例，并且相对于普通的正例和负例来说，它们对学习可能更有用。与此相关的思想是主动学习(active learning)，其中学习算法能够自己生成实例，并要求被标记，而不像在其他学习算法中那样被动地被给定(Angluin 1988)(参见习题6)。 36

VC维在20世纪70年代初由Vapnik和Chervonenkis提出，新近的相关资源是Vapnik 1995，其中他指出“没有什么比好的理论更实用”。像在其他科学领域一样，这也在机器学

习领域得到了证实。你不必急于使用计算机；你可以使用纸张、铅笔，也许还需要橡皮擦之类的东西来思考，节省自己的时间，避免无用的编程。

PAC 模型由 Valiant(1984)提出，对于学习矩形的 PAC 分析来自 Blumer 等(1989)。一本涵盖 PAC 学习及 VC 维的计算学习理论的好教材是 Kearns 和 Vazirani(1994)。

2.10 习题

1. 编程，实现由给定训练集求解 S 和 G。
2. 设想你一次只能得到一个训练实例而不是一次得到所有的训练实例，请问这种情况下如何增量地调整 S 和 G？(提示：参考 Mitchell 1997 中的候选排除算法。)
3. 为什么使用 S 和 G 的平均值作为最终假设比较好？
4. 假定我们的假设类是圆而不是矩形。参数是什么？这种情况下如何计算圆假设的参数？如果是椭圆又如何？为什么用椭圆代替圆会更有意义？如何将算法推广到 $K>2$ 个类情况？
5. 设想我们的假设类不是一个矩形而是两个(或 $m>1$ 个)矩形的并，请问这种假设类优点何在？说明使用足够大的 m，任何类都能够由这种假设类表示。
6. 如果我们拥有能够给任何实例 x 提供标记的指导者，那么我们应当在哪里选择 x，以便用较少的询问来进行学习？
7. 在(2.12)式中，我们对实际值与估计值之差的平方求和。该误差函数是使用最频繁的误差函数，但它只是可行的误差函数之一。由于它对差的平方求和，所以它对于离群点不是鲁棒的。为了实现鲁棒回归(robust regression)，更好的误差函数是什么？
8. 请推导(2.16)式。
9. 假定我们的假设类是直线的集合，并且我们利用直线来分开正例与负例，而不是用矩形来界定正例，并将负例留在矩形外(参见图 2-10)。证明直线的 VC 维为 3。

37

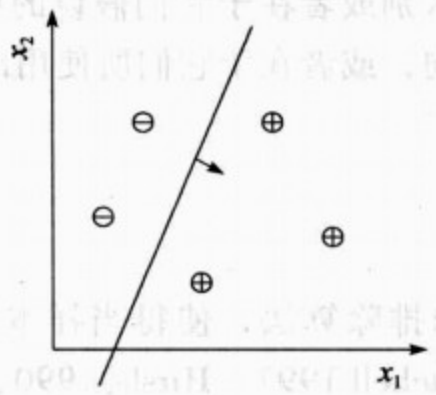


图 2-10 直线分割正例样本与负例样本

10. 证明在二维空间中，三角形假设类的 VC 维为 7。(提示：为了最佳分割，最好在某圆上设置 7 个等距离的点。)

2.11 参考文献

Angluin, D. 1988. "Queries and Concept Learning." *Machine Learning* 2: 319 – 342.

Blumer, A., A. Ehrenfeucht, D. Haussler, and M. K. Warmuth. 1989. "Learnability and the Vapnik-Chervonenk-

is Dimension." *Journal of the ACM* 36: 929 – 965.

Dietterich, T. G. 2003. "Machine Learning." In *Nature Encyclopedia of Cognitive Science*. London: Macmillan.

Hirsh, H. 1990. *Incremental Version Space Merging: A General Framework for Concept Learning*. Boston: Kluwer.

Kearns, M. J., and U. V. Vazirani. 1994. *An Introduction to Computational Learning Theory*. Cambridge, MA: The MIT Press.

Mitchell, T. 1997. *Machine Learning*. New York: McGraw-Hill.

Valiant, L. 1984. "A Theory of the Learnable." *Communications of the ACM* 27: 1134 – 1142.

Vapnik, V. N. 1995. *The Nature of Statistical Learning Theory*. New York: Springer.

Winston, P. H. 1975. "Learning Structural Descriptions from Examples." In *The Psychology of Computer Vision*, ed. P. H. Winston, 157 – 209. New York: McGraw-Hill.

38

第3章　贝叶斯决策定理

我们讨论在不确定情况下决策的概率理论框架。在分类中，贝叶斯规则用来计算类的概率。我们将讨论推广到怎样做出合理的决策将期望风险最小化。我们还介绍贝叶斯网络来有效地表示随机变量之间的依赖关系。

3.1　引言

训练计算机使之根据数据进行推断是统计学和计算机科学的交叉领域，其中，统计学家提供由数据推断的数学框架，而计算机科学家研究推断方法在计算机上如何有效地实现。

数据来自于一个不完全清楚的过程。将该过程作为随机过程建模表明我们缺乏知识。也许该过程是确定性的，但是因为我们没有获取关于它的完全知识的途径，所以我们把它作为一个随机过程建模，并且用概率理论来分析它。说到这，在继续阅读本章之前先跳到附录，温习基本的概率知识也许是一个好主意。

投硬币是一个随机的过程，因为我们不能够预测任意一次投币结果是正面还是反面（这就是为什么我们投币、买彩票或者买保险的原因）。我们只能谈论下一次投币是正面还是反面的概率。有证据显示，如果我们取得一些额外的数据，如硬币的确切成分，它的最初

39 位置，投币的力量和投币的方向，何处以及如何接住等等，则投币的准确结果就是可以预测的。

我们不能获取的那些额外的数据称为不可观测的变量（unobservable variable）。在投币的这个例子中，唯一可观测的变量（observable variable）是投币的结果。用 z 表示不可观测的变量，x 表示可观测的变量，事实上我们有

$$x = f(z)$$

其中，$f(\cdot)$ 是一个确定性函数，它定义不可观测数据的输出。因为我们不能用这种方式对该过程建模，所以我们定义输出 X 为指明该过程、由概率分布 $P(X=x)$ 抽取的随机变量。

投币的结果是正面或是反面，而我们定义一个随机变量，在两个值中取值。令 $X=1$ 代表投币的结果是正面，$X=0$ 代表投币结果是反面。X 服从伯努利分布，其中参数 p_0 是投币结果为正面的概率。

$$P(X=1)=p_0 \text{ 并且 } P(X=0)=1-P(X=1)=1-p_0$$

假设要预测下一次投币的结果。如果我们知道 p_0 的值，则当 $p_0>0.5$ 时，预测将是正面，否则是反面。这是因为，如果选择更可能的情况，则错误的概率，即 1 减去选择的概率，将会最小。如果这是一个 $p_0=0.5$ 的公平投币，则我们没有比总是选择正面或者我们自己做公平投币更好的预测手段！

如果我们不知道 $P(X)$，并且想从给定的样本估计它，就需要统计学知识了。我们有一

40 个样本 $\mathcal{X}$，包含由可观测变量 x^t 的概率分布（记作 $p(x)$）抽取出的样例。目的是使用样本 $\mathcal{X}$

构造一个它的近似 $\hat{p}(x)$。

在投币例子中，样本包含了 N 次投币的结果。然后利用 $\boldsymbol{\chi}$，我们可以估计 p_0。p_0 是唯一定义该分布的参数。p_0 的估计是

$$\hat{p}_0 = \frac{\#\{\text{结果为正面的投币}\}}{\#\{\text{投币}\}}$$

使用数值随机变量，如果投币 t 的结果是正面，则 x^t 为 1，否则为 0。给定样本{正面，正面，正面，反面，正面，反面，反面，正面，正面}，则 $X = \{1, 1, 1, 0, 1, 0, 0, 1, 1\}$，并且 40

$$\hat{p}_0 = \frac{\sum_{t=1}^{N} x^t}{N} = \frac{6}{9}$$

3.2 分类

在 1.2.2 节，我们讨论了信用评分。在那里我们看到，根据过去的交易，银行的某些客户是低风险的，因为他们还清了贷款并且银行从他们那里获利；其他客户是高风险的，因为他们不能偿还贷款。分析这些数据，我们想学习“高风险客户”类，使得未来有新的贷款申请时，我们可以检查申请者是否符合“高风险客户”类的描述，并据此决定接受还是拒绝该申请。使用关于申请的知识，我们假定有两种信息是可观测的。我们观测它们是因为我们有理由相信它们为我们提供了客户信用信息。例如，我们假定我们观测客户的年收入和存款，它们分别用随机变量 X_1 和 X_2 表示。

可以断言，如果我们能够获得客户的其他知识，比如关于客户经济状况的全部细节和全部知识，他/她的意图、道德规范等等，则我们可以确定地计算出客户是“低风险客户”还是“高风险客户”。但是，这些是不可观测的，而使用我们能够观测的信息，客户的信用可以用观测条件 $\boldsymbol{X} = [X_1, X_2]^T$ 下的伯努利随机变量 C 表示，其中 $C = 1$ 表示高风险客户，$C = 0$ 表示低风险客户。如果我们知道 $P(C \mid X_1, X_2)$，则当一个 $X_1 = x_1$ 和 $X_2 = x_2$ 的新申请到达时，我们可以

$$\text{选择}\begin{cases} C = 1 & \text{如果 } P(C = 1 \mid x_1, x_2) > 0.5 \\ C = 0 & \text{否则} \end{cases}$$

或等价地

$$\text{选择}\begin{cases} C = 1 & \text{如果 } P(C = 1 \mid x_1, x_2) > P(C = 0 \mid x_1, x_2) \\ C = 0 & \text{否则} \end{cases} \tag{3.1}$$

错误的概率是 $1 - \max(P(C = 1 \mid x_1, x_2), P(C = 0 \mid x_1, x_2))$。这个例子与抛硬币的例子类似，只是伯努利随机变量 C 是在两个其他观测变量条件下的随机变量。我们用 $\boldsymbol{x}$ 表示观测变量向量 $\boldsymbol{x} = [x_1, x_2]^T$。于是，问题是要能够计算 $P(C \mid \boldsymbol{x})$。使用贝叶斯规则，它可以表示为 41

$$p(C \mid \boldsymbol{x}) = \frac{P(C)p(\boldsymbol{x} \mid C)}{p(\boldsymbol{x})} \tag{3.2}$$

$P(C = 1)$ 称为 C 取值 1 的先验概率(prior probability)。在我们的例子中，它对应客户是高风险客户的概率，而不管 $\boldsymbol{x}$ 取什么值。称它为先验概率，是因为它是我们看到观测量 $\boldsymbol{x}$ 之

前就获得的关于 C 值的知识，满足

$$P(C=0)+P(C=1)=1$$

$p(\boldsymbol{x}\mid C)$ 称为类似然(class likelihood)，是属于 C 的事件具有相关联的观测值 $\boldsymbol{x}$ 的条件概率。在我们的例子中，$p(x_1, x_2 \mid C=1)$ 是高风险客户具有 $X_1=x_1$，$X_2=x_2$ 的概率。这就是通过数据我们得到的关于类的信息。

$p(\boldsymbol{x})$ 是证据(evidence)，是看到观测 $\boldsymbol{x}$ 的边缘概率，无论它是正实例还是负实例。

$$p(\boldsymbol{x})=p(\boldsymbol{x}\mid C=1)P(C=1)+p(\boldsymbol{x}\mid C=0)P(C=0) \tag{3.3}$$

使用贝叶斯规则，组合先验知识和数据告诉我们的，在看到观测 $\boldsymbol{x}$ 之后，计算概念的后验概率(posterior probability) $P(C\mid\boldsymbol{x})$。

$$\text{后验}=\frac{\text{先验}\times\text{似然值}}{\text{证据}}$$

由于用证据规范化，后验的和为1：

$$P(C=0\mid\boldsymbol{x})+P(C=1\mid\boldsymbol{x})=1$$

一旦得到后验概率，我们就可以使用(3.1)式进行决策。从现在起，我们假定我们知道先验和似然。在稍后的章节中，我们会讨论如何从训练样本估计 $P(C)$ 和 $p(\boldsymbol{x}\mid C)$。

在一般情况下，我们有 K 个互斥和穷举的类 C_i，$i=1,\cdots,K$；例如，在光学数字识别中，输入是一个位图图像，有10个类。我们得到先验概率满足：

42

$$P(C_i)\geqslant 0 \text{ 并且 } \sum_{i=1}^{K}P(C_i)=1 \tag{3.4}$$

$p(\boldsymbol{x}\mid C_i)$ 是已知属于类 C_i，看到 $\boldsymbol{x}$ 作为输入的概率。类 C_i 的后验概率计算如下

$$P(C_i\mid\boldsymbol{x})=\frac{p(\boldsymbol{x}\mid C_i)P(C_i)}{p(\boldsymbol{x})}=\frac{p(\boldsymbol{x}\mid C_i)P(C_i)}{\sum_{k=1}^{K}p(\boldsymbol{x}\mid C_k)P(C_k)} \tag{3.5}$$

而为了将误差最小化，贝叶斯分类器(Bayes' classifier)选择具有最高后验概率的类；即

$$\text{选择 } C_i \text{ 如果 } P(C_i\mid\boldsymbol{x})=\max_k P(C_k\mid\boldsymbol{x}) \tag{3.6}$$

3.3 损失与风险

决策的好坏程度或代价可能不同。金融机构对一个贷款申请人做出决定时会把潜在的收益和损失考虑在内。接受一个低风险的申请人会增加收益，而拒绝一个高风险的申请人会减小损失。错误地接受一个高风险的申请人带来的损失与错误地拒绝一个低风险的申请人带来的潜在收益是不同的。这种情况在其他领域，如医疗诊断、地震预测等，显得更加至关重要并且是非常不对称的。

让我们定义动作 α_i 为把输入指派到类 C_i 的决策，而 λ_{ik} 为输入实际属于 C_k 时采取动作 α_i 导致的损失(loss)。采取动作 α_i 带来的期望风险(expected risk)是

$$R(\alpha_i\mid\boldsymbol{x})=\sum_{k=1}^{K}\lambda_{ik}P(C_k\mid\boldsymbol{x}) \tag{3.7}$$

并且我们选择具有最小风险的动作：

$$\text{选择 } \alpha_i \text{ 如果 } R(\alpha_i\mid\boldsymbol{x})=\min_k R(\alpha_k\mid\boldsymbol{x}) \tag{3.8}$$

让我们定义 K 个动作 α_i，$i=1$，…，K，其中 α_i 是把 $\boldsymbol{x}$ 指派到 C_i 的动作。在 0-1 损失(zero-one loss)这种特殊情况下，其中

$$\lambda_{ik} = \begin{cases} 0 & \text{如果 } i = k \\ 1 & \text{如果 } i \neq k \end{cases} \tag{3.9}$$

所有正确的决策没有损失，并且所有错误具有相同的代价。采取动作 α_i 的风险是

$$\begin{aligned} R(\alpha_i \mid \boldsymbol{x}) &= \sum_{k=1}^{K} \lambda_{ik} P(C_k \mid \boldsymbol{x}) \\ &= \sum_{k \neq i} P(C_k \mid \boldsymbol{x}) \\ &= 1 - P(C_i \mid \boldsymbol{x}) \end{aligned}$$

43

因为 $\sum_k P(C_k \mid \boldsymbol{x}) = 1$。所以为了将风险最小化，我们选择最有可能的情况。在后面章节中，为了简单起见，我们一直假定这种情况，并且选择具有最高后验的类，但是注意，这确实是一种特殊情况，并且很少应用具有对称的 0-1 损失。在一般情况下，由后验到风险并且采取将风险最小化的动作是一种简单的后处理。

在一些应用中，错误的决策(即误分类)也许会有很高的代价。一般情况下，如果自动系统对它的决策的把握较低，则需要一个更复杂(例如人工的)决策。例如，如果我们使用光学数字识别器来读取信封上的邮编号码，则错误地识别邮政编码将导致信件被发送到一个错误的目的地。

在这种情况下，我们定义一个附加的拒绝(reject)或疑惑 (doubt) 动作 α_{K+1}，而 $\alpha_i (i = 1, \cdots, K)$ 是在类 $C_i (i = 1, \cdots, K)$ 上的通常的决策动作(Duda、Hart 和 Stork 2001)。

一个可能的损失函数是

$$\lambda_{ik} = \begin{cases} 0 & \text{如果 } i = k \\ \lambda & \text{如果 } i = K+1 \\ 1 & \text{否则} \end{cases} \tag{3.10}$$

其中 $0 < \lambda < 1$ 是选择第 $(K+1)$ 个拒绝动作导致的损失。拒绝的风险是

$$R(\alpha_{K+1} \mid \boldsymbol{x}) = \sum_{k=1}^{K} \lambda P(C_k \mid \boldsymbol{x}) = \lambda \tag{3.11}$$

而选择类 C_i 的风险是

$$R(\alpha_i \mid \boldsymbol{x}) = \sum_{k \neq i} P(C_k \mid \boldsymbol{x}) = 1 - P(C_i \mid \boldsymbol{x}) \tag{3.12}$$

最优决策规则是

选择 C_i　如果对于所有的 $k \neq i$ 有 $R(\alpha_i \mid \boldsymbol{x}) < R(\alpha_k \mid \boldsymbol{x})$，并且 $R(\alpha_i \mid \boldsymbol{x}) < R(\alpha_{K+1} \mid \boldsymbol{x})$

拒绝　如果 $R(\alpha_{K+1} \mid \boldsymbol{x}) < R(\alpha_i \mid \boldsymbol{x}), i = 1, \cdots, K$ (3.13)

44

给定(3.10)式的损失函数，上式可以简化为

选择 C_i　如果对于所有的 $k \neq i$ 有 $P(C_i \mid \boldsymbol{x}) > P(C_K \mid \boldsymbol{x})$，并且 $P(C_i \mid \boldsymbol{x}) > 1 - \lambda$

拒绝　否则 (3.14)

当$0<\lambda<1$时，这个方法是有意义的：当$\lambda=0$时，我们总是拒绝；拒绝和正确的分类是同样好的。当$\lambda\geqslant 1$时，我们从不拒绝；拒绝与错误的代价相同甚至超过错误的代价。

3.4 判别式函数

分类也可以看作是实现一组判别式函数(discriminant function)$g_i(\boldsymbol{x})(i=1,\cdots,K)$使得我们

$$\text{选择 } C_i,\text{如果 } g_i(\boldsymbol{x})=\max_k g_k(\boldsymbol{x}) \tag{3.15}$$

令

$$g_i(\boldsymbol{x})=-R(\alpha_i\mid\boldsymbol{x})$$

我们可以重新给出贝叶斯分类器，并且最大化判别式函数对应最小化条件风险。当我们使用0-1损失函数时，我们有

$$g_i(\boldsymbol{x})=P(C_i\mid\boldsymbol{x})$$

或者忽略公共规范化项$p(\boldsymbol{x})$，我们可以写为

$$g_i(\boldsymbol{x})=p(\boldsymbol{x}\mid C_i)P(C_i)$$

这把特征空间划分成K个决策区域(decision region)$\mathfrak{R}_1,\cdots,\mathfrak{R}_K$，其中$R_i=\{\boldsymbol{x}\mid g_i(\boldsymbol{x})=\max_k g_k(\boldsymbol{x})\}$。这些区域被决策边界(decision boundaries)，即特征空间中的曲面分隔开，其中平局出现在最大判别式函数之间(参见图3-1)。

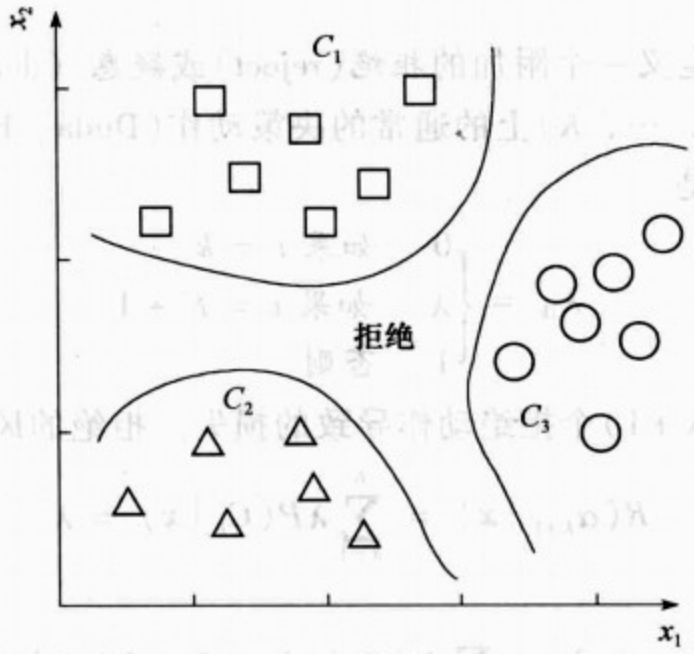


图3-1 决策区域和决策边界的例子

当有两个类的时候，我们可以定义单个判别式

$$g(\boldsymbol{x})=g_1(\boldsymbol{x})-g_2(\boldsymbol{x})$$

并且我们

$$\text{选择}\begin{cases}C_1 & \text{如果 } g(\boldsymbol{x})>0\\ C_2 & \text{否则}\end{cases}$$

一个例子是两类学习问题，其中正例可以表示为C_1，负例表示为C_2。当$K=2$时，分类系统是一个两分器(dichotomizer)，当$K\geqslant 3$时，它是一个多分器(polychotomizer)。

3.5 效用理论

在(3.7)式中，我们定义了期望风险并且选择最小化期望风险的动作。现在，我们把它推广到效用理论，它涉及我们对状态不确定时所做出的合理决策。我们假设给定证据 $\boldsymbol{x}$，状态 S_k 的概率用 $P(S_k \mid \boldsymbol{x})$ 计算。我们定义一个效用函数(utility function) U_{ik}，它度量当状态为 S_k 时采取动作 α_k 的好处。期望效用(expected utility)是

$$EU(\alpha_i \mid \boldsymbol{x}) = \sum_k U_{ik} P(S_k \mid \boldsymbol{x}) \tag{3.16}$$

一个合理的决策者选择最大化期望效用的动作

$$\text{选择 } \alpha_i, \text{如果 } EU(\alpha_i \mid \boldsymbol{x}) = \max_j EU(\alpha_j \mid \boldsymbol{x}) \tag{3.17}$$

对于分类，决策对应选择类中的一个，并且最大化期望效用等价于最小化期望风险。U_{ik} 一般用货币术语度量，并且这也为我们提供了一种定义损失矩阵 λ_{ik} 的途径。例如，在我们定义拒绝选项(3.10 式)时，根据我们的特定应用，如果我们知道一个正确决策能给我们带来多少钱的收益，一个错误决策将使我们损失多少钱，还有把决策委托给人类专家的代价多大，那么我们就能用货币单位，而不是用 0、λ 和 1 填写 U_{ik} 的正确值，并且做出我们的决策来最大化期望收益。 46

对于拒绝，我们在计算机程序做出的自动决策和代价较大但是正确可能性更高的人类决策之间选择。类似地，可以设想一条多个自动决策者的流水线，我们处理起来代价比较高，但是正确的机会更大。

注意，最大化期望效用只是一种可能的方法；我们也可以定义其他类型的合理行为，例如，最小化最坏的可能损失。

3.6 信息值

在医疗诊断中，对一个病人可以有多种检查。测量脉搏没有开销，但是验血的开销很大，不仅是化验费用，而且给病人带来不便。然而，验血可以为我们提供更多信息。一般来说，我们假定所有的可观测特征都会被观测，但情况并非总是如此。某些特征，如医疗诊断中的验血，观测的开销很大，并且仅当我们确实需要时才进行观测。因此，我们希望能够评估附加特征可能提供的信息。

假设我们有已经观测的特征 $\boldsymbol{x}$，那么当前最好动作的期望效用是

$$EU(\boldsymbol{x}) = \max_i \sum_k U_{ik} P(S_k \mid \boldsymbol{x})$$

如果我们观测了新的特征 z，并将它与 $\boldsymbol{x}$ 一起使用，则当前最好动作的期望效用是

$$EU(\boldsymbol{x}, z) = \max_i \sum_k U_{ik} P(S_k \mid \boldsymbol{x}, z)$$

如果 $EU(\boldsymbol{x}, z) > EU(\boldsymbol{x})$，则我们可以说 z 是有用的，并且它们的差是 z 提供的信息值(value of information)。但是我们也应该考虑观测和处理 z 的开销：$P(S_k \mid \boldsymbol{x}, z)$ 使用 $\boldsymbol{x}$ 和 z 两者，并且比 $P(S_k \mid \boldsymbol{x})$ 更复杂。仅当 z 的贡献比它的附加的复杂度更值得的时候，我们才会把它作为新特征引入。 47

3.7 贝叶斯网络

贝叶斯网络(Bayesian network)，又称信念网络(belief network)或概率网络(probabilistic network)，是表示变量之间相互作用的图形模型(graphical model)。一个贝叶斯网络由节点和节点之间的弧组成。每个节点对应一个随机变量 X，并且具有一个对应该随机变量的概率值 $P(X)$。如果存在一条从节点 X 到节点 Y 的有向弧，则表明 X 对 Y 有直接影响(direct influence)。该影响被条件概率 $P(Y \mid X)$ 所指定。网络是一个有向无环图(directed acyclic graph)，即图中没有环。节点和节点之间的弧定义了网络的结构，而条件概率是给定结构的参数。

一个简单的例子由图 3-2 给出，它对下雨(R)引起草地变湿(W)建模。天下雨的可能性为 40%，并且下雨时草地变湿的可能性为 90%；也许 10% 的时间雨下得不长，不足以让我们真正认为草地被淋湿了。在这个例子中，随机变量是二元的：真或假。存在 20% 的可能
48 性草地变湿而实际上并没有下雨，例如，使用喷水器时。

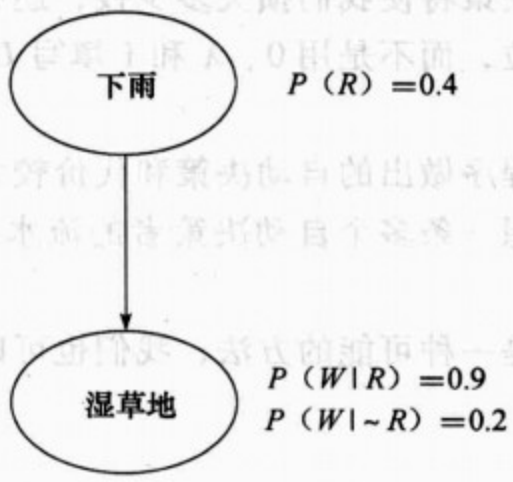


图 3-2 对下雨是湿草地的原因建模的贝叶斯网络

我们可以看到三个值就可以完全指定 $P(R, W)$ 的联合分布。如果 $P(R) = 0.4$，则 $P(\sim R) = 0.6$。类似地，$P(\sim W \mid R) = 0.1$，而 $P(\sim W \mid \sim R) = 0.8$。

这是一个因果图(causal graph)，解释草地变湿的主要原因是下雨。贝叶斯法则允许我们颠倒因果关系并且做出诊断(diagnosis)。例如，已知草地是湿的，则下过雨的概率可以计算如下：

$$
\begin{aligned}
P(R \mid W) &= \frac{P(W \mid R)P(R)}{P(W)} \\
&= \frac{P(W \mid R)P(R)}{P(W \mid R)P(R) + P(W \mid \sim R)P(\sim R)} \\
&= \frac{0.9 \times 0.4}{0.9 \times 0.4 + 0.2 \times 0.6} = 0.75
\end{aligned}
$$

分母 $P(W)$ 是草地变湿的概率，不管是否下过雨。注意，已知草地是湿的把下雨的概率由 0.4 增加到 0.75，这是因为 $P(W \mid R)$ 高，而 $P(W \mid \sim R)$ 低。

现在，假设我们想把喷水器(S)作为草地变湿的另一个原因，如图 3-3 所示。节点 W 有两个父节点 R 和 S，因此它的概率是这两个值上的条件概率 $P(W \mid R, S)$。我们可以计算喷
49 水器开着草地会湿的概率，无须知道是否下过雨。这是一个因果(预测)推理：

$$P(W|S) = P(W|R,S)P(R|S) + P(W|\sim R,S)P(\sim R|S)$$
$$= P(W|R,S)P(R) + P(W|\sim R,S)P(\sim R)$$
$$= 0.95 \times 0.4 + 0.9 \times 0.6 = 0.92$$

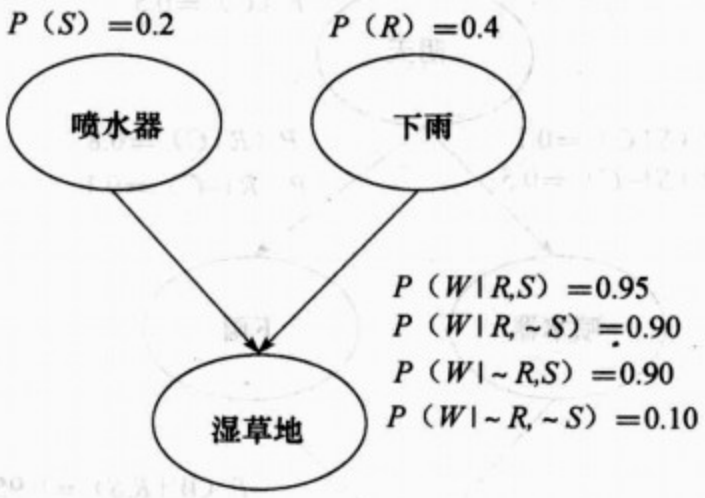


图 3-3 下雨和喷水器是草地变湿的两个原因

其中 $P(R|S) = P(R)$，因为根据图 3-3，R 和 S 是独立的。给定草地是湿的，我们能够计算喷水器开着的概率。这是一个诊断推理

$$P(S|W) = \frac{P(W|S)P(S)}{P(W)} = \frac{0.92 \times 0.2}{0.52} = 0.35$$

其中

$$P(W) = P(W|R,S)P(R,S) + P(W|\sim R,S)P(\sim R,S)$$
$$+ P(W|R,\sim S)P(R,\sim S) + P(W|\sim R,\sim S)P(\sim R,\sim S)$$
$$= P(W|R,S)P(R)P(S) + P(W|\sim R,S)P(\sim R)P(S)$$
$$+ P(W|R,\sim S)P(R)P(\sim S) + P(W|\sim R,\sim S)P(\sim R)P(\sim S)$$
$$= 0.95 \times 0.4 \times 0.2 + 0.9 \times 0.6 \times 0.2 + 0.9 \times 0.4 \times 0.8 + 0.1 \times 0.6 \times 0.8$$
$$= 0.52$$

知道草是湿的增加了喷水器开着的可能。现在让我们假设下过雨，我们有

$$P(S|R,W) = \frac{P(W|R,S)P(S|R)}{P(W|R)} = \frac{P(W|R,S)P(S)}{P(W|R)} = 0.21$$

注意，这个值比 $P(S|W)$ 小。这叫作解释远离（explaining away）；给定我们已知下过雨，则喷水器导致湿草地的可能性降低了。已知草地是湿的，下雨和喷水器成为相互依赖的。

图 3-3 表示 R 和 S 是互相独立的。然而，我们可以认为它们实际上依赖于另外一个变量的出现：如果可能下雨的话，我们通常不把喷水器打开。所以，一个更好的图在图 3-4 给出。如果是阴天，有可能会下雨，我们可能会发现喷水器是关着的。例如，我们能够计算如果是阴天，草地湿的概率：

$$P(W|C) = P(W|R,S,C)P(R,S|C) + P(W|\sim R,S,C)P(\sim R,S|C)$$
$$+ P(W|R,\sim S,C)P(R,\sim S|C) + P(W|\sim R,\sim S,C)P(\sim R,\sim S|C)$$
$$= P(W|R,S)P(R|C)P(S|C) + P(W|\sim R,S)P(\sim R|C)P(S|C)$$
$$+ P(W|R,\sim S)P(R|C)P(\sim S|C)$$
$$+ P(W|\sim R,\sim S)P(\sim R|C)P(\sim S|C)$$

其中，给定 R，S，W 独立于 C，我们使用了 $P(W \mid R, S, C) = P(W \mid R, S)$。类似地，给定 C，R 和 S 是独立的，$P(R, S \mid C) = P(R \mid C)P(S \mid C)$。这是贝叶斯网络的优点，它明确地表示了独立性，并且使得我们能够将推断分解成若干变量小组上的计算。

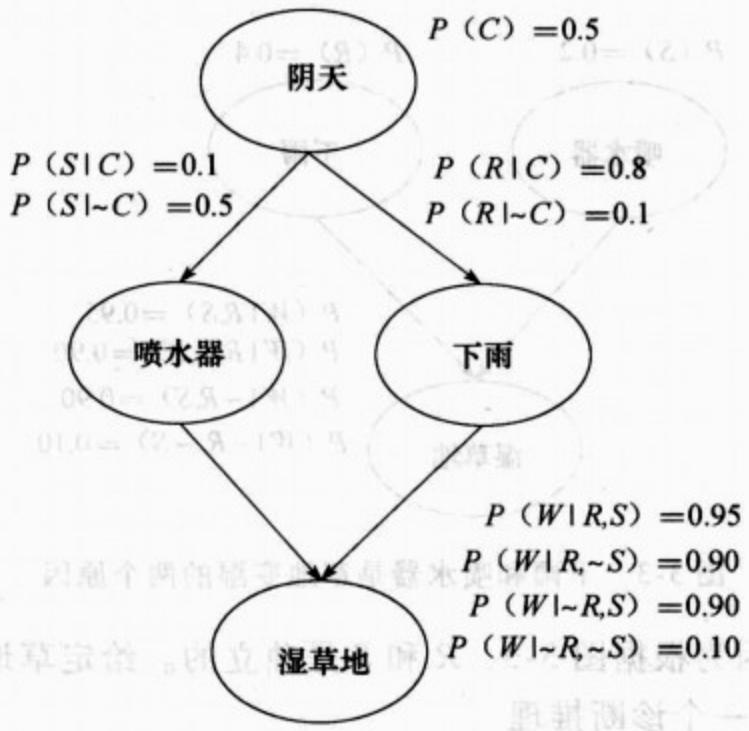


图 3-4 如果阴天，可能会下雨，我们就不会使用喷水器

根据我们能够观测到的，我们可以使网络更加详细。例如，我们也许有一只猫，它喜欢在房顶上走动并发出噪声(即使房顶不是铁皮的)；下雨时猫不出去(图 3-5)。于是，例如给

51 定阴天，我们能够计算听见猫在房顶上走动(F)的概率 $P(F \mid C)$，甚至计算 $P(F \mid S)$。

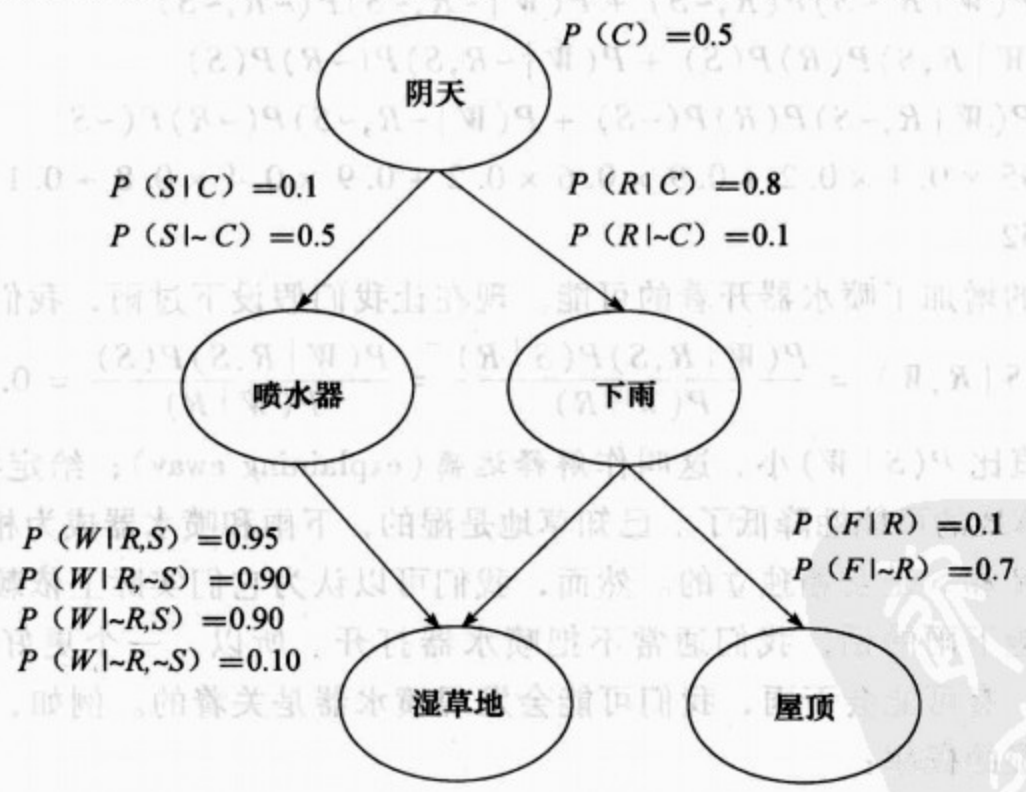


图 3-5 下雨不仅能够使草地变湿，而且会打扰经常在房顶发出噪声的猫

这种图形表示很形象，并且有助于理解。这个网络描述了条件独立性，并且允许我们将多个变量的联合分布的问题分解成局部(local)结构；这简化了分析和计算。图 3-5 表示了一

52 个五个二元变量的联合密度，它通常需要存储 31 个值($2^5 - 1$)，然而这里只有 11 个。如果

每个节点只有少量的父节点，则复杂度由指数降到线性(按节点数)。正如我们在前面所看到的，当联合密度分解成较小变量组的条件密度时，推断也会变得更容易：

$$P(C,S,R,W,F) = P(C)P(S|C)P(R|C)P(W|S,R)P(F|R) \tag{3.18}$$

虽然在这个例子中我们用的是二元变量，但是显然这些变量可以是具有任意多个可能值的离散变量，或者是连续变量。这只是改变了条件概率。在一般情况下，当我们有变量 $X_1, \cdots, X_d$ 时，

$$P(X_1,\cdots,X_d) = \prod_{i=1}^{d} P(X_i \mid \text{parents}(X_i)) \tag{3.19}$$

给定 X_i 的任意子集，即根据证据赋予它们一定的值，通过边缘化联合分布，我们可以计算 X_i 的其他子集的概率分布。这开销很大，因为它需要计算指数多个联合概率组合，即使每个都能像(3.18)式那样被简化。存在一种称作信念传播(belief propagation)的有效的算法(Pearl 1988)，当网络是树时，我们可以用它来推断。还有一个算法，通过聚类变量，把给定的有向无环图转化成一棵树，称作结树(junction tree)，使得信念传播能够完成(Lauritzen 和 Spiegelhalter 1988)。

使用贝叶斯网络的一个主要优点是我们不必明确指定某些变量作为输入，某些其他变量作为输出。任何变量集的值都能通过证据建立，而任何其他变量集的概率都可以推断，并且非监督学习和监督学习之间的区别变得模糊不清。

需要强调一点，从节点 X 到 Y 的链不是，也不必总是意味因果关系(causality)。它只是意味着在 Y 的概率以 X 的值为条件的意义下，X 在 Y 上有直接影响(direct influence)，并且即使没有直接的原因，两个节点之间仍可能有一个边。通过提供关于数据如何产生的解释，在构造网络时有因果关系更可取(Pearl 2000)，但是这种因果关系并非总是能够得到的。

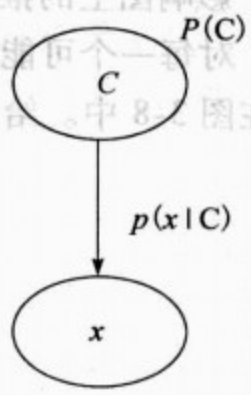


图 3-6 用于分类的贝叶斯网络

我们在本书中讨论的大部分方法都可以表示成贝叶斯网络。例如，对于我们在 3.2 节中讨论的分类，对应的贝叶斯网络如图 3-6 所示。(3.2)式给出的贝叶斯规则可以用来计算 $p(C|\boldsymbol{x})$，即诊断。

如果输入是相互独立的，如图 3-7 所示，该网络叫做朴素贝叶斯分类器(naive Bayes' classifier)，因为它忽略了输入之间的可能的依赖性(即相关性)，将一个多变量问题归约为一组单变量问题：

$$p(\boldsymbol{x}|C_i) = \prod_{j=1}^{d} p(x_j|C_i)$$

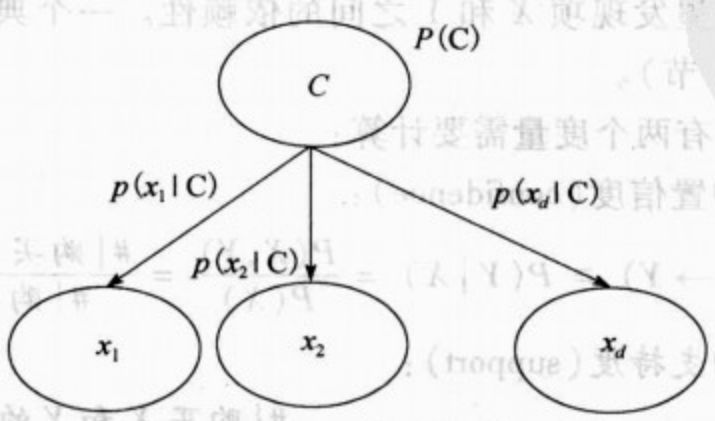


图 3-7 朴素贝叶斯分类器是一个假设输入变量相互独立的用于分类的贝叶斯网络

在一个问题中，可能存在隐藏变量(hidden variable)，它们的值不能通过证据知道。使用隐藏变量的优点是可以更容易地定义依赖结构。例如，在购物篮分析，当我们想找出所销售的商品之间的依赖性时，比如说我们知道“婴儿食品”、“尿布”和“牛奶”之间的依赖性，因为顾客购买其中一种多半可能购买另外两种。我们不是将(非因果的)弧放在这三者
54 之间，而可能是指定一个隐藏节点“家庭婴儿”作为这三种商品消费的隐藏原因。当存在隐藏节点时，它们的值用观测节点的给定值估计并填写。

用来学习贝叶斯网络参数的学习算法与我们将在后面章节中讨论的方法并无差别(Buntine 1996)。它们基本上是估计条件概率。学习结构要困难得多。尽管已经为此提出了一些算法，但是基本上都是人类专家来定义变量之间的因果关系，并创建小变量组的层次结构。

3.8 影响图

正如我们前面将概率推广到具有风险的动作一样，影响图(influence diagram)是一种图形模型，是包括决策和效用的贝叶斯网络的推广。一个影响图包含机会节点(chance node)，表示我们在贝叶斯网络中使用的随机变量。影响图还包含决策节点和效用节点。决策节点(decision node)表示动作的选择。效用节点(utility node)是效用被计算的地方。决策可以根据机会节点做出，还可能影响其他机会节点和效用节点。

影响图上的推断是对贝叶斯网络上的推断的扩展。给定一些机会节点上的证据，证据被传播，对每一个可能的决策计算效用，并选择具有最大效用的决策。对一个给定输入分类的影响图在图3-8中。给定输入，决策节点决定类，对于每个选择，我们得到一定的效用(风险)。

55 图3-8 对应分类的影响图。依赖于输入 x，选择一个导致一定效用(风险)的类

3.9 关联规则

关联规则(association rule)是形如 $X \rightarrow Y$ 的蕴涵式。关联规则的一个例子是购物篮分析，通过购物篮分析，我们希望发现项 X 和 Y 之间的依赖性。一个典型的应用是零售，其中 X 和 Y 是销售的商品(1.2.1节)。

在学习关联规则时，有两个度量需要计算：

- 关联规则 $X \rightarrow Y$ 的置信度(confidence)：

$$\text{Confidence}(X \rightarrow Y) \equiv P(Y \mid X) = \frac{P(X,Y)}{P(X)} = \frac{\#\{\text{购买 } X \text{ 和 } Y \text{ 的顾客}\}}{\#\{\text{购买 } X \text{ 的顾客}\}} \tag{3.20}$$

- 关联规则 $X \rightarrow Y$ 的支持度(support)：

$$\text{Support}(X,Y) \equiv P(X,Y) = \frac{\#\{\text{购买 } X \text{ 和 } Y \text{ 的顾客}\}}{\#\{\text{顾客}\}} \tag{3.21}$$

置信度是我们通常计算的条件概率 $P(Y|X)$。为了能够说该规则具有足够的置信度，它的值应该接近于 1，并且显著大于人们购买 Y 的总体概率 $P(Y)$。我们也对最大化规则的支持度感兴趣，因为即使有一个强置信度的依赖，如果这样的顾客数量很少，该规则也是没有价值的。支持度展示了规则的统计显著性，而置信度展示了规则的强度。

最小的支持度和置信度由公司设定，并且在数据库中搜索所有具有较高支持度和置信度的规则。支持度和置信度的公式可以很容易推广到多于两个项，使得 X 和 Y 是不相交的项集。例如，$P(Y|X, Z)$是三项集，比两项集更重要。由于销售数据库一般非常大，所以我们希望通过少数几遍数据库扫描找出依赖性。有一个有效的算法，称作 Apriori 算法(Agrawal 等 1996)，来做这项工作。

56

3.10 注释

依据不确定数据进行决策已经有很长的历史，并且人类一直在探索各种陌生领域，寻找证据来排除不确定性：例如天体、水晶球和咖啡杯。使用概率论，从有意义的证据推理仅有几百年的历史。关于概率和统计学的历史，以及拉普拉斯、伯努利和创建该理论的其他学者的一些早期论文见 Newman1988。

Russell 和 Norving(1995)给出了效用理论和信息值的很好讨论，该书还用货币术语讨论了效用分配。Shafer 和 Pearl 1986 是不确定下推理的早期论文的汇集。Pearl 1988 年的书是一部经典之作，而他最近的书(Pearl 2000)更详细地研究了因果关系的概念。Jensen 1996 年的书是一本值得一读的贝叶斯网络导论。Lauritzen 1996 给出了更形式化的处理。Huang 和 Darwiche(1994)是一篇详细介绍结树的建立和在树上推理的好文章。当网络盛行的时候，精确推理变得不可行；可以使用随机抽样(Andrieu 等 2003)或者使用变通的方法(Jordan 等 1999)。Buntine1996 包含了在贝叶斯网络中学习的文献评论，Jordan1999 是更多最近工作的汇编。Cowell 等 1999 讨论了通过专家以及通过从数据中学习结构的方法建立网络结构。

人工智能方面与不确定性有关的一个网页在 http://www.auai.org，那里提供了精选的引导性论文和到公共领域程序的链接。贝叶斯网络的近期工作可以在人工智能不确定性(Uncertainty in Artificial Intelligence, UAI)论文集中找到。

3.11 习题

1. 在两个类的问题中，似然比(likelihood ratio)是

$$\frac{p(x|C_1)}{p(x|C_2)}$$

请用似然比写出判别函数。

2. 在两个类的问题中，对数几率(log odd)定义为

$$\log \frac{P(C_1|x)}{P(C_2|x)}$$

请用对数几率写出判别函数。

3. 在两类、两动作问题中，如果损失函数是 $\lambda_{11}=\lambda_{22}=0$，$\lambda_{12}=10$，$\lambda_{21}=1$，写出最佳决策规则。

57

4. 某人做公平投币，如果结果是正面，你什么都得不到，否则就会得到 $5。玩这样的游戏你愿意支付多少钱？如果赢 $500 而不是 $5又如何？
5. 在图 3-4 中，计算 $P(C \mid W)$。
6. 在图 3-5 中，计算 $P(F \mid C)$。
7. 给定图 3-5 的结构和包含下表所示观测的样本 X，如何学习概率？

阴天	喷水器	下雨	湿草地	屋顶
否	是	否	是	是
是	否	是	否	否
…	…	…	…	…

8. 推广购物篮分析的置信度和支持度公式，计算 k-依赖，即 $P(Y \mid X_1, \cdots, X_k)$。
9. 在购物篮分析中，如果对于每件售出的商品我们还有一个数，该数指出顾客喜爱该商品的程度，例如，在 0 到 10 这个范围内，你怎么能利用这一附加信息计算把哪种商品推荐给顾客？

3.12 参考文献

Agrawal, R., H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. 1996. "Fast Discovery of Association Rules." In *Advances in Knowledge Discovery and Data Mining*, ed. U. M. Fayyad, G. Piatetsky- Shapiro, P. Smyth, and R. Uthurusamy, 307 – 328. Cambridge, MA: The MIT Press.

Andrieu, C., N. de Freitas, A. Doucet, and M. I. Jordan. 2003. "An Introduction to MCMC for Machine Learning." *Machine Learning* 50: 5 – 43.

Buntine, W. 1996. "A Guide to the Literature on Learning Probabilistic Networks from Data." IEEE *Transactions on Knowledge and Data Engineering* 8: 195 – 210.

Cowell, R. G., A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. New York: Springer.

58

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, 2nd ed. New York: Wiley.

Huang, C., and A. Darwiche. 1994. "Inference In Belief Networks: A Procedural Guide." *International Journal of Approximate Reasoning* 11: 1 – 158.

Jensen, F. 1996. *An Introduction to Bayesian Networks*. New York: Springer.

Jordan, M. I., ed. 1999. *Learning in Graphical Models*. Cambridge, MA: The MIT Press.

Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. 1999. "An Introduction to Variational Methods for Graphical Models." In *Learning in Graphical Models*, ed. M. I. Jordan, 105 – 161. Cambridge, MA: The MIT Press.

Lauritzen, S. L. 1996. *Graphical Models*. Oxford: Oxford University Press.

Lauritzen, S. L., and D. J. Spiegelhalter. 1988. "Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems." *Journal of Royal Statistical Society Series* B 50: 157 – 224.

Newman, J. R., ed. 1988. *The World of Mathematics*. Redmond, WA: Tempus.

Pearl, J. 1988. *Probabilistic Reasoning in Expert Systems*. San Mateo, CA: Morgan Kaufmann.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge, UK: Cambridge University Press.

Russell, S., and P. Norvig. 1995. *Artificial Intelligence: A Modern Approach*. New York: Prentice Hall.

59

Shafer, G., and J. Pearl, eds. 1990. *Readings in Uncertain Reasoning*. San Mateo, CA: Morgan Kaufmann.

第4章 参数方法

前面，我们讨论了在使用概率对不确定性建模时，如何做出最优决策。现在，考虑如何从给定的训练集估计这些概率。我们从分类和回归的参数方法开始。在后面的章节中，将讨论半参数和非参数方法，介绍用于权衡模型复杂度和经验误差的模型选择方法。

4.1 引言

统计量(statistic)是从给定样本中计算的任何值。在统计推断中，我们使用样本提供的信息进行决策。第一种方法是参数方法，这里假设样本取自服从已知模型的某个分布，例如高斯分布。参数方法的优点是，模型定义在少量参数(例如均值、方差)，即分布的有效统计量上。一旦从样本中估计出这些参数，就知道了整个分布。我们从给定的样本估计分布的参数，把这些估计放到假设的模型中，并得到估计的分布，然后用它进行决策。我们用来估计分布参数的方法是最大似然估计。我们还将讨论贝叶斯估计，随着计算能力的提高，贝叶斯估计日趋流行。

我们从密度估计(density estimation)开始。密度估计是估计 $p(x)$ 的一般情况。我们使用密度估计进行分类，其中估计的密度是能够计算后验概率 $P(C_i \mid x)$ 并做决策的类密度 $p(x \mid C_i)$ 和 $P(C_i)$。然后，我们讨论回归，其中估计的密度是 $p(y \mid x)$。本章，x 是一维的，因此密度是一元的。在第5章中，我们将推广到多元情况。

60
~
61

4.2 最大似然估计

假定我们有一个独立同分布(iid)样本 $\mathcal{X} = \{x^t\}_{t=1}^N$。我们假设 x^t 是从某个定义在参数 θ 上的已知概率密度族 $p(x \mid \theta)$ 中抽取的实例：

$$x^t \sim p(x \mid \theta)$$

我们希望找出这样的 θ，使得 x^t 尽可能像是从 $p(x \mid \theta)$ 抽取的。因为 x^t 是独立的，给定参数 θ，样本 $\mathcal{X}$ 的似然(likelihood)是个体点似然的乘积：

$$l(\mathcal{X} \mid \theta) \equiv p(\mathcal{X} \mid \theta) = \prod_{t=1}^{N} p(x^t \mid \theta) \tag{4.1}$$

在最大似然估计(maximum likelihood estimation)中，我们感兴趣的是找到这样的 θ，使得 $\mathcal{X}$ 最像是抽取的。因此，我们寻找最大化样本似然的 θ，该似然记作 $l(\theta \mid \mathcal{X})$。我们可以最大化该似然的对数，而不改变它取最大值的数值。$\log(\cdot)$ 把乘积转换为求和，并且当假定某种密度(例如包含指数)时进一步简化计算量。对数似然(log likelihood)定义为

$$\mathcal{L}(\theta \mid \mathcal{X}) \equiv \log l(\theta \mid \mathcal{X}) = \sum_{t=1}^{N} \log p(x^t \mid \theta) \tag{4.2}$$

现在，让我们来看我们感兴趣的实际应用中出现的一些分布。如果我们有两类问题，

62

我们就用伯努利分布。当存在 $K>2$ 个类时，分布拓广为多项分布。高斯(正态)密度是一种最常使用的对类条件输入建模的密度。对于这三种分布，我们讨论它们的参数的最大似然估计(MLE)。

4.2.1 伯努利密度

在伯努利分布中，有两个结果：事件要么发生，要么不发生。例如，实例是类的正例，或者不是。事件发生，伯努利随机变量 X 以概率 p 取值1，事件不发生的概率为 $1-p$，并用 X 取值0表示。这表示为

$$P(x) = p^x(1-p)^{1-x}, \quad x \in \{0,1\} \tag{4.3}$$

p 是唯一参数，并且给定独立同分布样本 $\mathcal{X}=\{x^t\}_{t=1}^N$，其中 $x^t \in \{0, 1\}$，我们希望计算 p 的估计 $\hat{p}$。对数似然是

$$\begin{aligned}\mathcal{L}(p|\mathcal{X}) &= \log\prod_{t=1}^{N} p^{(x^t)}(1-p)^{(1-x^t)} \\ &= \sum_t x^t \log p + \left(N - \sum_t x^t\right)\log(1-p)\end{aligned}$$

通过求解 $dL/dp=0$ 可以找出最大化该对数似然的 $\hat{p}$。

$$\hat{p} = \frac{\Sigma_t x^t}{N} \tag{4.4}$$

p 的估计是事件发生的次数与试验次数的比值。注意，如果 X 是参数为 p 的伯努利变量，则 $E[X]=p$，并且作为期望，均值的最大似然估计是样本的平均值。

4.2.2 多项密度

以伯努利分布的推广为例，其中随机事件的结果不是两种状态，而是 K 种互斥、穷举状态之一(例如类)，其中每种状态出现的概率为 p_i，其中 $\Sigma_{i=1}^K p_i = 1$。设 x_1，x_2，…，x_K 是指示变量，其中当输出为状态 i 时 x_i 为1，否则为0。

$$P(x_1, x_2 \cdots, x_K) = \prod_{i=1}^{K} p_i^{x_i} \tag{4.5}$$

假定我们做 N 次这样的独立试验，结果为 $\mathcal{X}=\{x^t\}_{t=1}^N$，其中，

63

$$x_i^t = \begin{cases} 1 & \text{如果试验 } t \text{ 选择状态 } i \\ 0 & \text{否则} \end{cases}$$

其中 $\Sigma_i x_i^t = 1$。p_i 的最大似然估计是

$$\hat{p}_i = \frac{\Sigma_t x_i^t}{N} \tag{4.6}$$

状态 i 的概率估计是结果为状态 i 的试验次数与试验总次数的比值。有两种方法可以获得这个估计：如果 x_i 是0/1，则可以认为它们是 K 个独立的伯努利试验。或者，我们可以写出对数似然并找出最大化它的 p_i(满足条件 $\Sigma_i p_i = 1$)。

4.2.3 高斯(正态)密度

X 是均值为 μ 方差为 σ^2 的高斯(正态)分布，记作 $\mathcal{N}(\mu, \sigma^2)$，如果它的密度函数为

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \quad -\infty < x < \infty \tag{4.7}$$

给定样本 $\mathcal{X} = \{x^t\}_{t=1}^N$，其中 $x^t \sim \mathcal{N}(\mu, \sigma^2)$，高斯样本的对数似然为

$$\mathcal{L}(\mu, \sigma | \mathcal{X}) = -\frac{N}{2}\log(2\pi) - N\log\sigma - \frac{\Sigma_t (x^t - \mu)^2}{2\sigma^2}$$

最大似然估计为

$$m = \frac{\Sigma_t x^t}{N} \tag{4.8}$$

$$s^2 = \frac{\Sigma_t (x^t - m)^2}{N}$$

我们根据通常的约定，用希腊字母表示总体参数，用罗马字母表示它们的样本估计。有时，帽(抑扬符号)也用来表示估计，例如 $\hat{\mu}$。

4.3 评价估计：偏倚和方差

令 $\mathcal{X}$ 是取自参数 θ 指定的总体上的样本，并令 $d = d(\mathcal{X})$ 是 θ 的一个估计。为了评估该估计的质量，我们可以度量它与 θ 有多大不同，即 $(d(\mathcal{X}) - \theta)^2$。但是因为它是一个随机变量(它依赖于样本)，我们需要对它在可能的 $\mathcal{X}$ 上取平均值，并考虑 $r(d, \theta)$，它是估计 d 的均方误差(mean square error)，定义为 64

$$r(d, \theta) = E[(d(\mathcal{X}) - \theta)^2] \tag{4.9}$$

估计的偏倚(bias)是

$$b_\theta(d) = E[d(\mathcal{X})] - \theta \tag{4.10}$$

如果对所有的 θ 值都有 $b_\theta(d) = 0$，则 d 是 θ 的无偏估计(unbiased estimator)。例如，如果 x^t 是从均值为 μ 的密度抽取出的，则样本平均值 m 是均值 μ 的一个无偏估计，因为

$$E(m) = E\left[\frac{\Sigma_t x^t}{N}\right] = \frac{1}{N}\sum_t E[x^t] = \frac{N\mu}{N} = \mu$$

这就意味着虽然在一个特定样本上，m 却可能与 μ 不同，但是如果我们取许多这样的样本 $\mathcal{X}_i$，并且估计许多 $m_i = m(\mathcal{X}_i)$，随着样本的增加，它们的平均值将逼近 μ。m 也是一个一致估计，也就是说，当 $N \to \infty$ 时，$\mathrm{Var}(m) \to 0$。

$$\mathrm{Var}(m) = \mathrm{Var}\left(\frac{\Sigma_t x^t}{N}\right) = \frac{1}{N^2}\sum_t \mathrm{Var}(x^t) = \frac{N\sigma^2}{N^2} = \frac{\sigma^2}{N}$$

随着样本中的点数 N 的增大，m 对 μ 的偏离变小。现在，让我们来检查 σ^2 的最大似然估计 s^2：

$$s^2 = \frac{\sum_t (x^t - m)^2}{N} = \frac{\sum_t (x^t)^2 - Nm^2}{N}$$

$$E[s^2] = \frac{\sum_t E[(x^t)^2] - N \cdot E[m^2]}{N}$$

给定 $\mathrm{Var}(X) = E[X^2] - E[X]^2$，我们得到 $E[X^2] = \mathrm{Var}(X) + E[X]^2$，并且

$$E[(x^t)^2] = \sigma^2 + \mu^2, \quad E[m^2] = \sigma^2/N + \mu^2$$

于是，我们有

$$E[s^2] = \frac{N(\sigma^2 + \mu^2) - N(\sigma^2/N + \mu^2)}{N} = \left(\frac{N-1}{N}\right)\sigma^2 \neq \sigma^2$$

上式说明 s^2 是 σ^2 的有偏估计。$(N/(N-1))s^2$ 是一个无偏估计。然而，当 N 很大时，差别可以忽略。这是一个渐进无偏估计(asymptotically unbiased estimator)的例子，它的偏倚随着 N 趋向无穷而趋向于0。

65

均方误差可以重新改写如下(d 是 $d(X)$的缩写)：

$$\begin{aligned} r(d,\theta) &= E[(d-\theta)^2] \\ &= E[(d - E[d] + E[d] - \theta)^2] \\ &= E[(d - E[d])^2 + (E[d] - \theta)^2 + 2(E[d] - \theta)(d - E[d])] \\ &= E[(d - E[d])^2] + E[(E[d] - \theta)^2] + 2E[(E[d] - \theta)(d - E[d])] \\ &= E[(d - E[d])^2] + (E[d] - \theta)^2 + 2(E[d] - \theta)E[d - E[d]] \\ &= \underbrace{E[(d - E[d])^2]}_{\text{方差}} + \underbrace{(E[d] - \theta)^2}_{\text{偏倚}^2} \end{aligned} \tag{4.11}$$

最后两式相等是因为 $E[d]$是常数，因此 $E[d] - \theta$ 也是一个常数，并且因为 $E[d - E[d]] = E[d] - E[d] = 0$。在(4.11)式中，第一项是*方差*(variance)，度量在平均情况下 d_i 在期望值附近的变化程度(从一个数据集到另一个数据集)；而第二项是*偏倚*(bias)，度量期望值偏离正确值 θ 的程度(参见图4-1)。于是，我们把误差写成方差和偏倚的平方之和：

$$r(d,\theta) = \mathrm{Var}(d) + (b_\theta(d))^2 \tag{4.12}$$

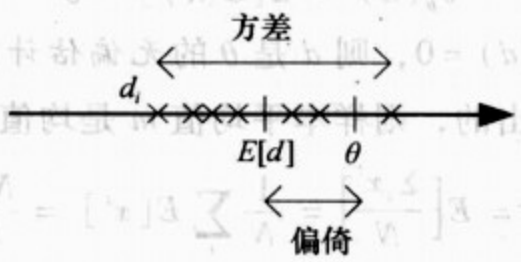


图4-1 θ 是需要估计的参数。d_i 是在不同样本上的多个估计(用“×”表示)。偏倚是 d 的期望值与 θ 之差。方差是 d_i 在期望值周围的散布程度。我们希望它们两个都很小

4.4 贝叶斯估计

有时，在看到样本之前，我们(或应用领域专家)可能会有一些关于参数 θ 可能取值的范围的先验(prior)信息。这些信息是非常有用的，也应当利用起来，尤其是样本较小时。

这些先验信息不会告诉我们参数的确切值(否则我们就不需要该样本),并且我们通过把 θ 看作是一个随机变量并为它定义先验密度 $p(\theta)$ 来对这种不确定性建模。例如,假设我们知道 θ 接近正态分布,并且 θ 介于 5 到 9 之间,7 左右对称的置信度为 90%。于是,可以把 $p(\theta)$ 写成均值为 7 的正态分布,并且因为

$$P\left\{-1.64 < \frac{\theta - \mu}{\sigma} < 1.64\right\} = 0.9$$

$$P\{\mu - 1.64\sigma < \theta < \mu + 1.64\sigma\} = 0.9$$

我们取 $1.64\sigma = 2$,并且使用 $\sigma = 2/1.64$。这样,我们就可以假定 $p(\theta) \sim \mathcal{N}(7, (2/1.64)^2)$。

先验密度(prior density)$p(\theta)$ 告诉我们在看到样本之前 θ 的可能取值。我们把它和样本数据告诉我们的(即似然密度 $p(\mathcal{X} \mid \theta)$)结合起来,利用贝叶斯规则,得到 θ 的后验密度(posterior density),它告诉我们看到样本之后 θ 的可能取值:

$$p(\theta \mid \mathcal{X}) = \frac{p(\mathcal{X} \mid \theta)p(\theta)}{p(\mathcal{X})} = \frac{p(\mathcal{X} \mid \theta)p(\theta)}{\int p(\mathcal{X} \mid \theta')p(\theta')d\theta'} \tag{4.13}$$

为了估计 x 上的密度,我们有

$$p(x \mid \mathcal{X}) = \int p(x, \theta \mid \mathcal{X})d\theta$$

$$= \int p(x \mid \theta, \mathcal{X})p(\theta \mid \mathcal{X})d\theta$$

$$= \int p(x \mid \theta)p(\theta \mid \mathcal{X})d\theta$$

$p(x \mid \theta, \mathcal{X}) = p(x \mid \theta)$,因为只要我们知道有效统计量 θ,我们就知道关于分布的一切。这样,我们在使用所有 θ 的值的预测上取平均值,用它们的概率加权。如果我们像在回归中一样,以 $y = g(x \mid \theta)$ 的形式做预测,则有

$$y = \int g(x \mid \theta)p(\theta \mid \mathcal{X})d\theta$$

66
~
67

除非后验具有很好的形式,否则求这个积分可能非常困难。当求整个积分不可行时,我们把它缩减到单个点。如果我们可以假定 $p(\theta \mid \mathcal{X})$ 在它的众数周围有一个窄的峰值,则使用最大后验(maximum a posteriori, MAP)估计将使得计算比较容易:

$$\theta_{MAP} = \arg\max_{\theta} p(\theta \mid \mathcal{X}) \tag{4.14}$$

这样,用单个点取代整个密度,回避积分并且使用

$$p(x \mid \mathcal{X}) = p(x \mid \theta_{MAP})$$

$$y_{MAP} = g(x \mid \theta_{MAP})$$

如果我们没有更重要的理由偏爱 θ 的某些值,则先验密度是扁平的,后验将与似然 $p(\mathcal{X} \mid \theta)$ 有同样的形式,并且 MAP 估计将等价于最大似然估计(参见 4.2 节),其中我们有

$$\theta_{ML} = \arg\max_{\theta} p(\mathcal{X} \mid \theta) \tag{4.15}$$

另外一个可能的方法是贝叶斯估计(Bayes' estimator)，它被定义为后验密度的期望值

$$\theta_{Bayes} = E[\theta|\mathcal{X}] = \int \theta p(\theta|\mathcal{X}) d\theta \tag{4.16}$$

取期望值的原因是随机变量的最佳估计是它的均值。假设 θ 是变量，我们想要用 $E[\theta]=\mu$ 预测。可以证明如果常数 c 是 θ 的估计，则

$$\begin{aligned} E[(\theta - c)^2] &= E[(\theta - \mu + \mu - c)^2] \\ &= E[(\theta - \mu)^2] + (\mu - c)^2 \end{aligned} \tag{4.17}$$

如果 c 取 μ，它的值最小。在正态密度情况下，众数是期望值。这样，如果 $p(\theta|\mathcal{X})$ 是正态的，则 $\theta_{Bayes} = \theta_{MAP}$。

作为一个例子，我们假设 $x^t \sim \mathcal{N}(\theta, \sigma_0^2)$ 并且 $\theta \sim \mathcal{N}(\mu, \sigma^2)$，其中 μ，σ 和 σ_0^2 已知：

$$p(\mathcal{X}|\theta) = \frac{1}{(2\pi)^{N/2}\sigma_0^N} \exp\left[-\frac{\Sigma_t (x^t - \theta)^2}{2\sigma_0^2}\right]$$

68

$$p(\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(\theta - \mu)^2}{2\sigma^2}\right]$$

可以证明 $p(\theta|\mathcal{X})$ 是正态的，满足

$$E[\theta|\mathcal{X}] = \frac{N/\sigma_0^2}{N/\sigma_0^2 + 1/\sigma^2} m + \frac{1/\sigma^2}{N/\sigma_0^2 + 1/\sigma^2} \mu \tag{4.18}$$

因此，贝叶斯估计是先验均值 μ 和样本均值 m 的加权平均值，权重与它们的方差成反比。利用样本提供的更多的信息，随着样本规模 N 的增加，贝叶斯估计逼近样本的平均值。当 σ^2 较小时，即当我们关于 θ 正确值具有较少的先验不确定性时，或者当 N 较小时，我们的先验猜测 μ 具有较好的效果。

注意，MAP 和贝叶斯估计都把整个后验密度归约到单个点并且丢失信息，除非后验是单模的并且在这些点周围有一个窄峰。随着计算费用降低，一个可能性是使用蒙特卡洛方法，它可以从后验密度产生样本(Andrieu 等 2003)。还有一些近似方法可以用来计算整个积分。

4.5 参数分类

我们在第 3 章看到，使用贝叶斯规则，我们可以把类 C_i 的后验概率写为

$$P(C_i|x) = \frac{p(x|C_i)P(C_i)}{p(x)} = \frac{p(x|C_i)P(C_i)}{\sum_{k=1}^{K} p(x|C_k)P(C_k)} \tag{4.19}$$

并使用判别式函数

$$g_i(x) = p(x|C_i)P(C_i)$$

或者等价地

$$g_i(x) = \log p(x|C_i) + \log P(C_i) \tag{4.20}$$

如果我们可以假设 $p(x|C_i)$ 是高斯的，则

$$p(x \mid C_i) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right] \tag{4.21}$$

(4.20)式变成

$$g_i(x) = -\frac{1}{2}\log 2\pi - \log \sigma_i - \frac{(x-\mu_i)^2}{2\sigma_i^2} + \log P(C_i) \tag{4.22}$$

69

让我们看一个例子：假设一个汽车公司销售 K 种不同的汽车，为了简单起见，我们假定唯一影响顾客购买的因素是他们的年收入，用 x 表示。于是，$P(C_i)$ 是购买类型 i 汽车的顾客所占的比例。如果顾客的年收入分布可以用一个高斯分布近似，则购买类型 i 汽车的顾客其收入为 x 的概率 $p(x \mid C_i)$ 服从分布 $\mathcal{N}(\mu_i, \sigma_i^2)$，其中 μ_i 是这类顾客年收入的均值，σ_i^2 是他们的年收入的方差。

当我们不知道 $P(C_i)$ 和 $p(x \mid C_i)$ 时，我们从样本估计它们并把它们的估计插入判别式，得到判别函数的估计。我们有样本

$$\mathcal{X} = \{x^t, \boldsymbol{r}^t\}_{t=1}^N \tag{4.23}$$

其中 $x \in \Re$ 是一维的，$\boldsymbol{r} \in \{0, 1\}^K$ 使得

$$r_i^t = \begin{cases} 1 & \text{如果 } x^t \in C_i \\ 0 & \text{如果 } x^t \in C_k, k \neq i \end{cases} \tag{4.24}$$

对于每一个类，均值和方差的估计是(依赖于 4.8 式)

$$m_i = \frac{\sum_t x^t r_i^t}{\sum_t r_i^t} \tag{4.25}$$

$$s_i^2 = \frac{\sum_t (x^t - m_i)^2 r_i^t}{\sum_t r_i^t} \tag{4.26}$$

而先验估计是(依赖于 4.6 式)

$$\hat{p}(C_i) = \frac{\sum_t r_i^t}{N} \tag{4.27}$$

把这些估计代入(4.22)式，得到

$$g_i(x) = -\frac{1}{2}\log 2\pi - \log s_i - \frac{(x-m_i)^2}{2s_i^2} + \log \hat{P}(C_i) \tag{4.28}$$

第一项是常数，可以去掉，因为它在所有的 $g_i(x)$ 中都是一样的。如果这些先验相等，则最后一项也可以去掉。如果我们进一步假设方差都相等，则上式可以写为

$$g_i(x) = -(x-m_i)^2 \tag{4.29}$$

因此我们把 x 指派到具有最近均值的类：

$$\text{选择 } C_i, \text{如果 } |x - m_i| = \min_k |x - m_k|$$

70

对于两个相邻的类，两个均值之间的中点是决策阈值(参见图 4-2)。

$$g_1(x) = g_2(x)$$

$$(x-m_1)^2 = (x-m_2)^2$$

$$x = \frac{m_1 + m_2}{2}$$

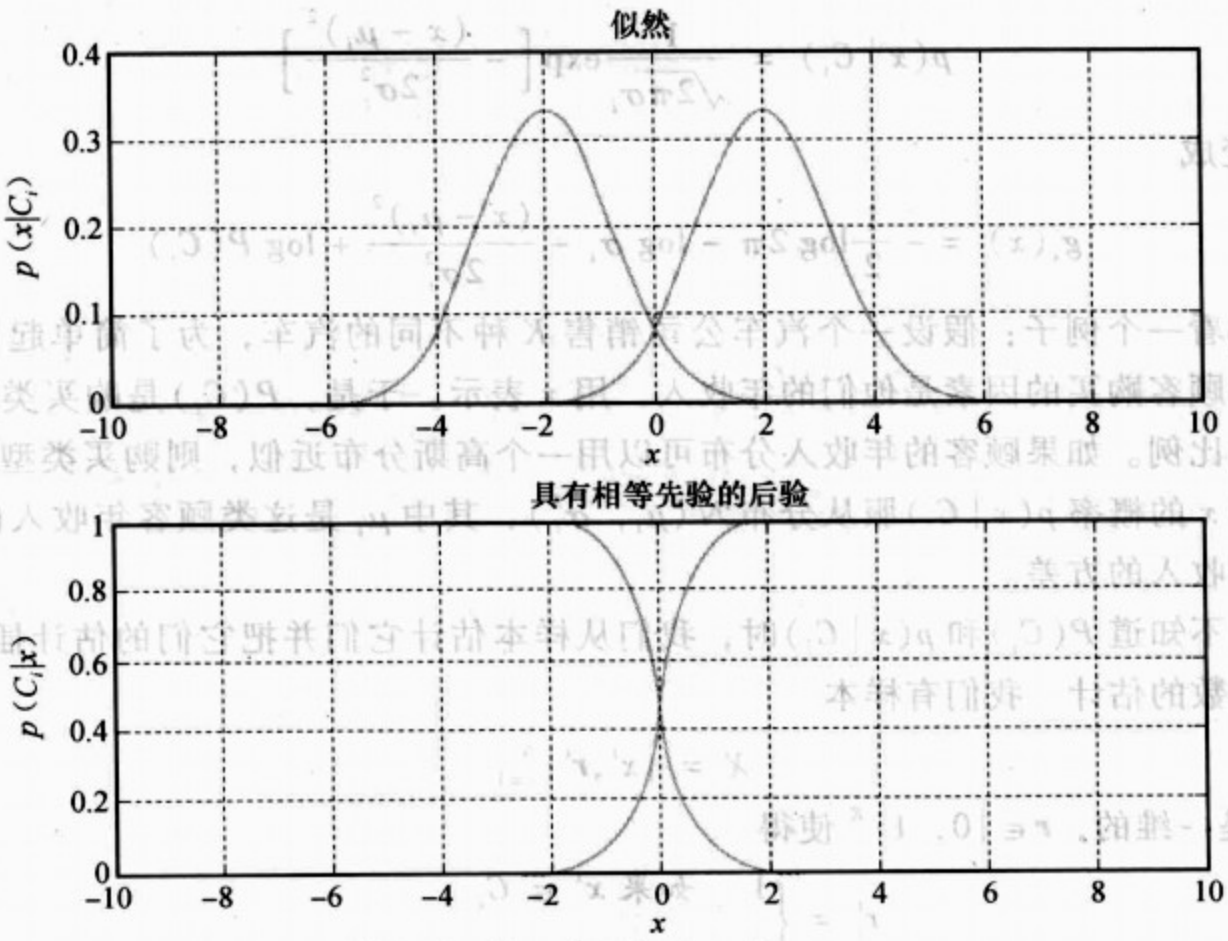


图 4-2 输入是一维的，似然函数和两个类具有相等先验的后验。
方差相等并且后验相交于一点，该点是决策阈值

当方差不相同时有两个阈值(参见图 4-3)，它们都容易计算(参见习题 4)。如果先验概率不同，则具有向不太可能的类的均值移动决策阈值的效果。

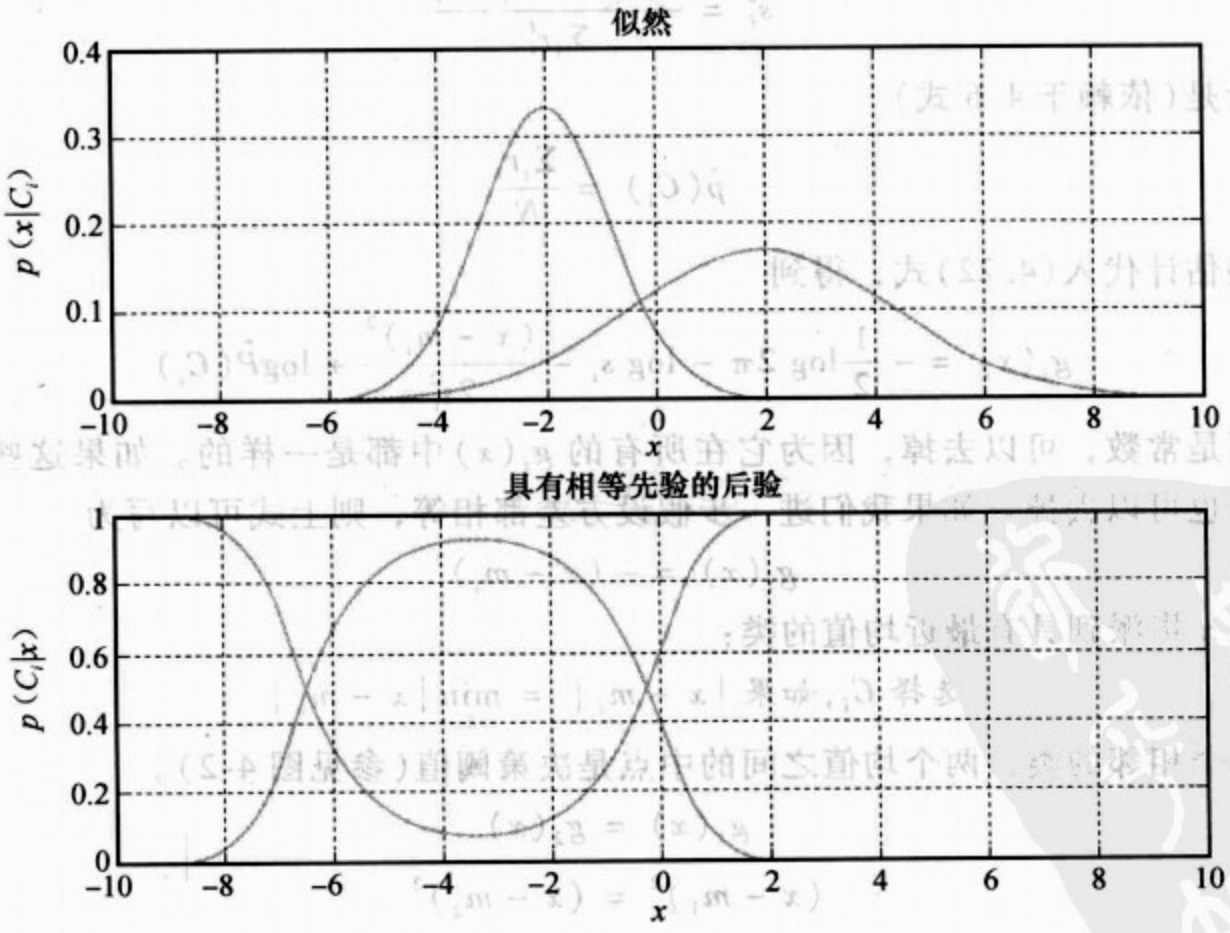


图 4-3 输入是一维的，似然函数和两个类具有相等先验的后验。
方差不相等并且后验在两个点上相交

这里，我们对参数使用最大似然估计。但是，如果有一些关于它们的先验信息(如均值)，则我们可以利用 μ_i 上的先验，使用 $p(x \mid C_i)$ 的贝叶斯估计。

这里必须注意：当 x 连续时，我们不要急于对 $p(x \mid C_i)$ 使用高斯密度。如果密度函数不是高斯的，则分类算法(即阈值点)将会出错。在统计学文献中，存在检查正态性的检验，并且这样的检验应该在假定正态分布之前使用。在一维数据的情况下，最简单的检验是绘制直方图并观察密度是否是钟形的，即是否是单峰并且围绕中心对称。

这是基于似然(likelihood-based approach)的分类方法，其中我们使用数据估计密度，使用贝叶斯规则计算后验密度，然后得到判别式。在稍后的章节中，我们讨论基于判别式的方法，那里我们将绕过密度估计而直接估计判别式。

71
~
72

4.6 回归

在回归中，我们喜欢将数值输出写成输入的函数。数值输出称为因变量(dependent variable)，函数的输入称为自变量(independent variable)。我们假定数值输出是输入的确定性函数与随机噪声的和：

$$r = f(x) + \varepsilon$$

其中 $f(x)$ 是未知函数，我们将用定义在参数 θ 的集合上的估计 $g(x \mid \theta)$ 来近似它。如果我们假设 ε 服从均值为 0，方差为 σ^2 的高斯分布，即 $\varepsilon \sim \mathcal{N}(0, \sigma^2)$，并且用我们的估计 $g(\cdot)$ 取代未知函数 $f(\cdot)$，则我们有(参见图 4-4)

$$p(r \mid x) \sim \mathcal{N}(g(x \mid \theta), \sigma^2) \tag{4.30}$$

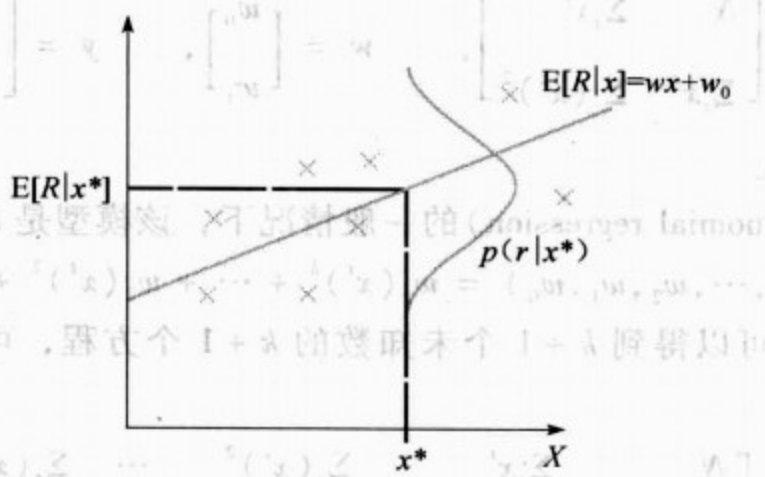


图 4-4 回归假定 0 均值的高斯噪声加到模型中，这里模型是线性的

我们再一次使用最大似然来学习参数 θ。训练集中的 (x^t, r^t) 对偶取自未知的联合概率密度 $p(x, r)$，可以写作

$$p(x, r) = p(r \mid x)p(x)$$

73

$p(r \mid x)$ 是在给定输入下输出的概率，而 $p(x)$ 是输入的密度。给定 iid 样本 $\mathcal{X} = \{x^t, r^t\}_{t=1}^N$，对数似然是

$$\mathcal{L}(\theta \mid \mathcal{X}) = \log \prod_{t=1}^{N} p(x^t, r^t) = \log \prod_{t=1}^{N} p(r^t \mid x^t) + \log \prod_{t=1}^{N} p(x^t)$$

我们可以忽略第二项，因为它不依赖于我们的估计。于是，我们有

$$\mathcal{L}(\theta \mid \mathcal{X}) = \log \prod_{t=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{[r^t - g(x^t \mid \theta)]^2}{2\sigma^2} \right]$$

$$= \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^N \exp\left[-\frac{1}{2\sigma^2} \sum_{t=1}^{N} [r^t - g(x^t \mid \theta)]^2 \right] \tag{4.31}$$

$$= -N \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{t=1}^{N} [r^t - g(x^t \mid \theta)]^2$$

第一项独立于参数 θ，可以去掉，因子 $1/\sigma^2$ 也可以去掉。最大化上式等价于最小化

$$E(\theta \mid \mathcal{X}) = \frac{1}{2} \sum_{t=1}^{N} [r^t - g(x^t \mid \theta)]^2 \tag{4.32}$$

它是我们最经常使用误差函数，而最小化它的 θ 叫做最小二乘估计(least squares estimate)。这是统计学经常做的一个变换：当似然 l 包含指数时，取代最小化 l，我们定义一个误差函数 $E = -\log l$，并最小化它。

在线性回归(linear regression)中，我们有线性模型

$$g(x^t \mid w_1, w_0) = w_1 x^t + w_0$$

取误差平方和的导数(4.32 式)，我们得到两个未知数的两个方程

$$\sum_t r^t = N w_0 + w_1 \sum_t x^t$$

$$\sum_t r^t x^t = w_0 \sum_t x_t + w_1 \sum_t (x^t)^2$$

它们可以被写成向量矩阵的形式 $\mathbf{A}\boldsymbol{w} = \boldsymbol{y}$，其中

$$\mathbf{A} = \begin{bmatrix} N & \sum_t x^t \\ \sum_t x^t & \sum_t (x^t)^2 \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} \sum_t r^t \\ \sum_t r^t x^t \end{bmatrix}$$

并且可以解得 $\boldsymbol{w} = \mathbf{A}^{-1}\boldsymbol{y}$。

在多项式回归(polynomial regression)的一般情况下，该模型是 k 次多项式

$$g(x^t \mid w_k, \cdots, w_2, w_1, w_0) = w_k (x^t)^k + \cdots + w_2 (x^t)^2 + w_1 x^t + w_0$$

并且取它的导数，我们可以得到 $k+1$ 个未知数的 $k+1$ 个方程，可以写做向量矩阵的形式 $\mathbf{A}\boldsymbol{w} = \boldsymbol{y}$，其中我们有

$$\mathbf{A} = \begin{bmatrix} N & \sum_t x^t & \sum_t (x^t)^2 & \cdots & \sum_t (x^t)^k \\ \sum_t x^t & \sum_t (x^t)^2 & \sum_t (x^t)^3 & \cdots & \sum_t (x^t)^{k+1} \\ \vdots & & & & \\ \sum_t (x^t)^k & \sum_t (x^t)^{k+1} & \sum_t (x^t)^{k+2} & \cdots & \sum_t (x^t)^{2k} \end{bmatrix}$$

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} \sum_t r^t \\ \sum_t r^t x^t \\ \sum_t r^t (x^t)^2 \\ \vdots \\ \sum_t r^t (x^t)^k \end{bmatrix}$$

记作 $\mathbf{A} = \mathbf{D}^T\mathbf{D}$ 和 $\boldsymbol{y} = \mathbf{D}^T\boldsymbol{r}$，其中

$$\mathbf{D} = \begin{bmatrix} 1 & x^1 & (x^1)^2 & \cdots & (x^1)^k \\ 1 & x^2 & (x^2)^2 & \cdots & (x^2)^k \\ \vdots & & & & \\ 1 & x^N & (x^N)^2 & \cdots & (x^N)^k \end{bmatrix}, \quad \boldsymbol{r} = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix}$$

然后，我们可以求解参数，得到

$$\boldsymbol{w} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\boldsymbol{r} \tag{4.33}$$

假设高斯分布误差，则最大化似然对应于最小化误差的平方和。另外一个度量是相对平方误差(relative square error，RSE)

$$E_{\text{RSE}} = \frac{\sum_t [r^t - g(x^t \mid \theta)]^2}{\sum_t (r^t - \bar{r})^2} \tag{4.34}$$

如果 E_{RSE} 接近于 1，则我们的预测与用平均值的预测一样好；当它接近于 0，我们得到更好的拟合。

记住，为了最好地泛化，我们应该调整学习模型的复杂度，以适应数据的复杂度。在多项式回归中，复杂度参数是拟合多项式的阶，因此需要找到一种选择最佳阶数的方法，能够 75
最小化泛化误差。也就是说，找到一种方法，调整模型的复杂度使其最佳拟合数据所固有的函数复杂度。

4.7 调整模型的复杂度：偏倚/方差两难选择

我们假设样本 $\mathcal{X} = \{x^t, r^t\}$ 取自未知的联合概率密度 $p(x, r)$。使用这个样本，我们构建估计 $g(\cdot)$。x 上的期望平方误差(联合密度上)可以表示为(用 4.17 式)

$$E[(r - g(x))^2 \mid x] = \underbrace{E[(r - E[r \mid x])^2 \mid x]}_{\text{噪声}} + \underbrace{(E[r \mid x] - g(x))^2}_{\text{误差的平方}} \tag{4.35}$$

右边的第一项是给定 x 时 r 的方差；它不依赖于 $g(\cdot)$ 或 $\mathcal{X}$。它是添加噪声的方差 σ^2。它是误差的一部分，无论我们使用什么估计方法，都不可能消除它。第二项量化 $g(x)$ 偏离回归函数 $E[r \mid x]$ 的程度。它确实依赖于估计方法和训练集。对一个样本来说，$g(x)$ 也许是一个非常好的拟合；而对其他样本，它可能是很差的拟合。为了评价一个估计 $g(\cdot)$ 的好坏程度，我们对可能的数据集进行平均。

期望值(样本 $\mathcal{X}$ 上的平均，所有的样本大小均为 N 并从相同联合密度 $p(x, r)$ 抽取)是(使用 4.11 式)

$$E_{\mathcal{X}}[(E[r \mid x] - g(x))^2 \mid x] = \underbrace{(E[r \mid x] - E_{\mathcal{X}}[g(x)])^2}_{\text{偏倚}} + \underbrace{E_{\mathcal{X}}[(g(x) - E_{\mathcal{X}}[g(x)])^2]}_{\text{方差}} \tag{4.36}$$

正如我们前面所讨论的，偏倚度量不考虑样本变化的影响时 $g(x)$ 的错误程度；方差度量当样本变化时 $g(x)$ 在期望值 $E[g(x)]$ 附近波动的程度。

让我们看一个例子：为了估计偏倚和方差，我们由某个带噪声的已知的 $f(\cdot)$ 产生一组数据集 $\mathcal{X}_i = \{x_i^t, r_i^t\}$，$i = 1, \cdots, M$，利用每个数据集形成一个估计 $g_i(\cdot)$，并计算偏倚和方差。注意，在现实生活中，我们不能够这么做，因为我们不知道 $f(\cdot)$，也不知道所添加噪声的参数。于是，$E[g(x)]$ 用 $g_i(\cdot)$ 上的平均来估计： 76

$$\bar{g}(x) = \frac{1}{M}\sum_{i=1}^{M} g_i(x)$$

偏倚和方差的估计是

$$\text{Bias}^2(g) = \frac{1}{N}\sum_t [\bar{g}(x^t) - f(x^t)]^2$$

$$\text{Variance}(g) = \frac{1}{NM}\sum_t \sum_i [g_i(x^t) - \bar{g}(x^t)]^2$$

让我们看几个不同复杂度的模型：最简单的是常数拟合

$$g_i(x) = 2$$

它没有方差，因为我们没有使用数据，并且所有的 $g_i(x)$ 都是相同的。但是偏倚很高，除非对于所有的 x，$f(x)$ 值都接近于 2。如果我们取样本中 r^t 的平均值

$$g_i(x) = \sum_t r_i^t / N$$

而不是常数 2，则就减少偏倚，因为我们预料在通常情况下，平均值是比常数更好的估计。但是，这增加了方差，因为不同的样本 $\mathcal{X}_i$ 将有不同的平均值。通常，在这种情况下，偏倚的减少比方差的增加更大，而误差将会降低。

图 4-5 给出了多项式回归的情况下的一个例子。随着多项式阶的增加，数据集的较小的变化将导致拟合多项式的较大变化；因此方差增加。但是，复杂的模型可以更好地拟合潜在的函数；因此偏倚减少(参见图 4-6)。这称为*偏倚/方差两难选择*(bias/variance dilemma)，并且不仅对于多项式回归，而且对于任何机器学习系统都存在这一问题(Geman、Bienenstock 和 Doursat 1992)。为了减少偏倚，冒着具有高方差的危险，模型应当是柔性的。如果方差保持较低，则我们可能不能很好地拟合数据，并且具有较高的偏倚。最佳模型是最好地权衡偏倚和方差的模型。

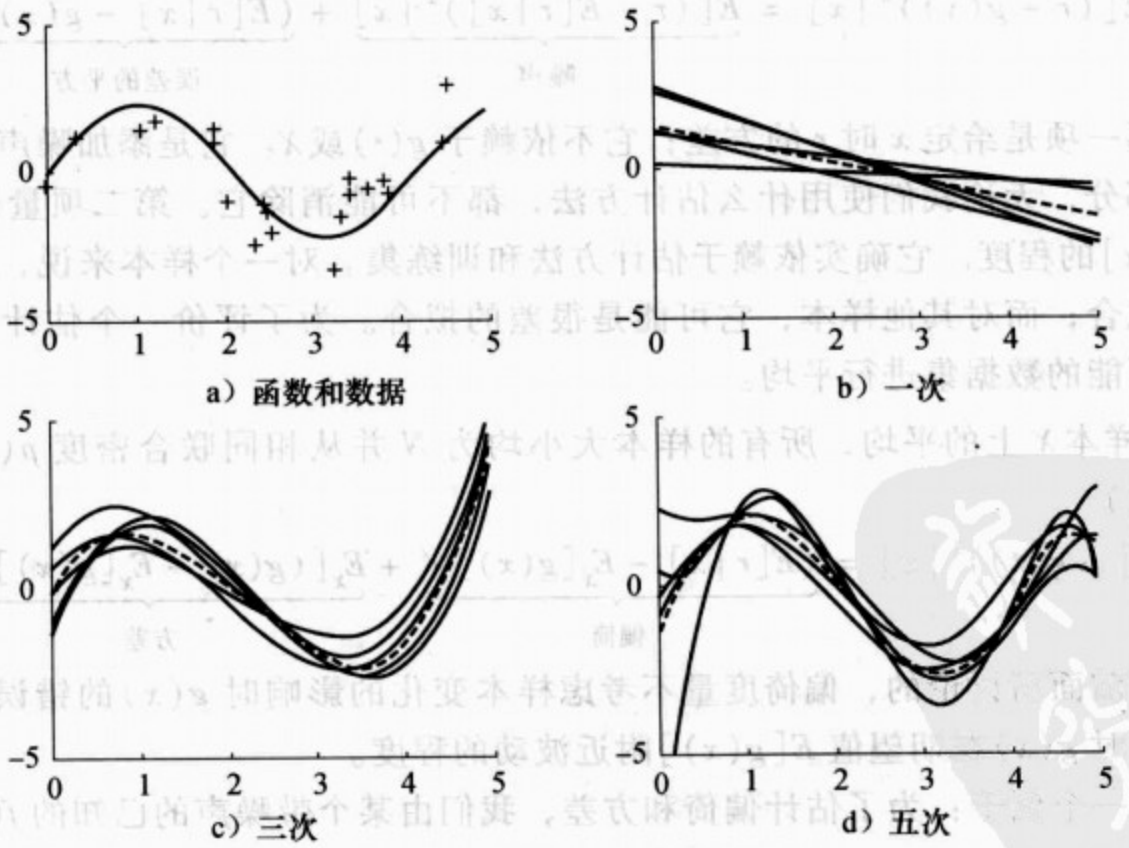


图 4-5 a)函数 $f(x) = 2\sin(1.5x)$ 和一个从该函数抽样含噪声($\mathcal{N}(0, 1)$)的数据集。抽取五个样本，每个包含 20 个实例。b)、c)、d)分别是 5 个一次、三次和五次多项式拟合，即 $g_i(\cdot)$。对每种情况，虚线是 5 次拟合的平均 $\bar{g}(\cdot)$

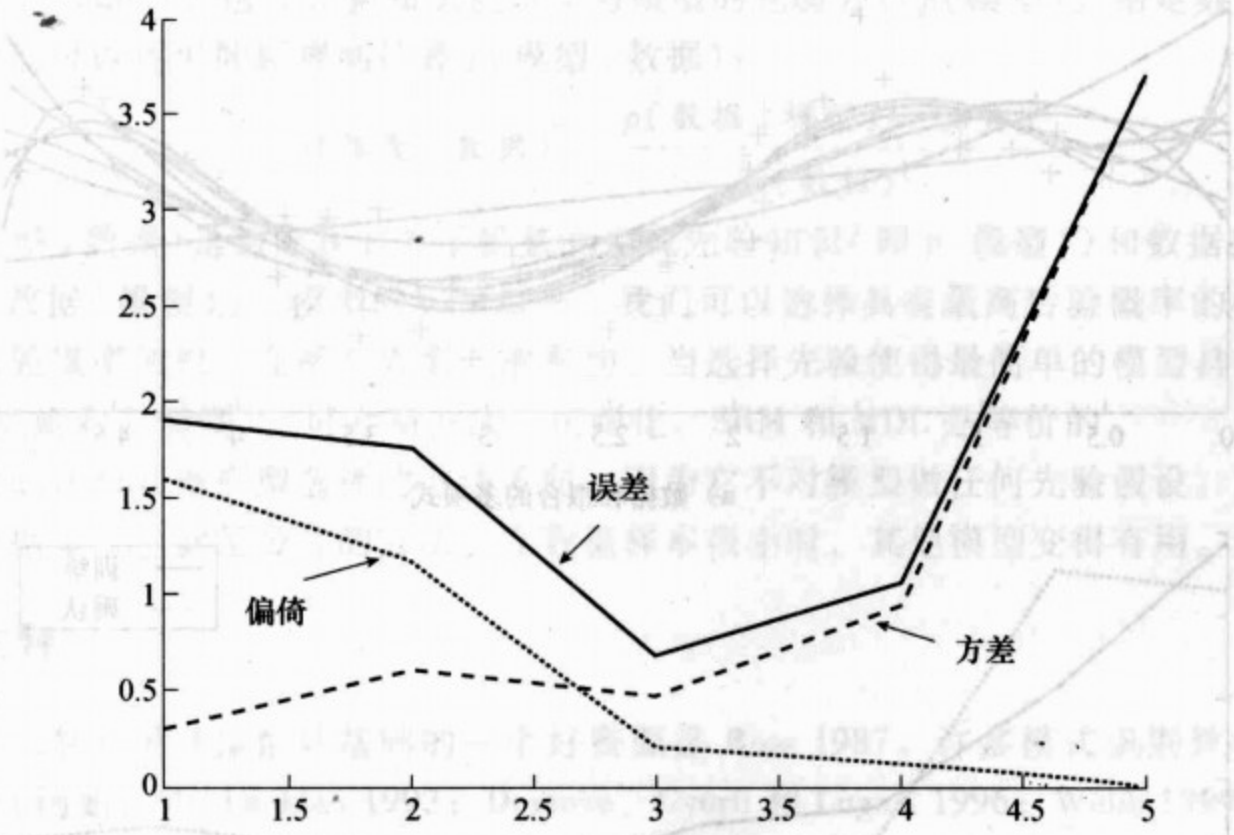


图 4-6 与图 4-5 同样的设置，使用 100 个模型而不是 5 个，从 1 到 5 次多项式的偏倚，方差和误差。一次多项式具有最小方差，5 次多项式具有最小偏倚。随着阶增加，偏倚减小但是方差增加。3 次多项式具有最小误差

如果有偏倚，这指明我们的模型类不包含解，这是欠拟合(underfitting)。如果有方差，则模型类过于一般，并且也学习噪声，这是过拟合(overfitting)。如果 $g(\cdot)$ 是与 $f(\cdot)$ 同样的假设类(例如同阶多项式)，则我们有一个无偏估计，并且估计的偏倚随着模型数量的增加而减小。这表明选择正确模型的误差降低效果(在第 2 章，我们称之为归纳偏倚——这两处“偏倚”的使用是不同的，但并非不相关)。对于方差，它同样依赖于训练集的大小；由于样本导致的可变性随着样本规模的增加而减少。总结一下，为了取得小的误差值，我们应该有合适的归纳偏倚(在统计意义上取得小偏倚)，并且有一个足够大的数据集，使得模型的可变性能够受到数据的约束。

注意，方差大时偏倚小，表明 $\bar{g}(x)$ 是一个好的估计器。因此，为了取得小误差，我们可以采用大量的高方差模型，并且用它们的平均值作为我们的估计。我们将在第 15 章讨论这种模型组合方法。

4.8 模型选择过程

有许多过程可以用来调整模型的复杂度。

在实践中，我们用来发现最佳复杂度的方法是交叉确认(cross-validation)。我们不能计算一个模型的偏倚和方差，但是我们能够计算总误差。给定一个数据集，我们把它分成两部分，分别作为训练集和确认集，在训练集上训练不同复杂度的候选模型，在确认集上测试它们的误差。随着模型复杂度增加，训练误差持续降低。在确认集的误差降低达到一定的复杂度水平之后，停止降低或不再明显的降低，如果有明显噪声的话甚至还会增加。这个“拐点”对应于最佳复杂度水平(参见图 4-7)。

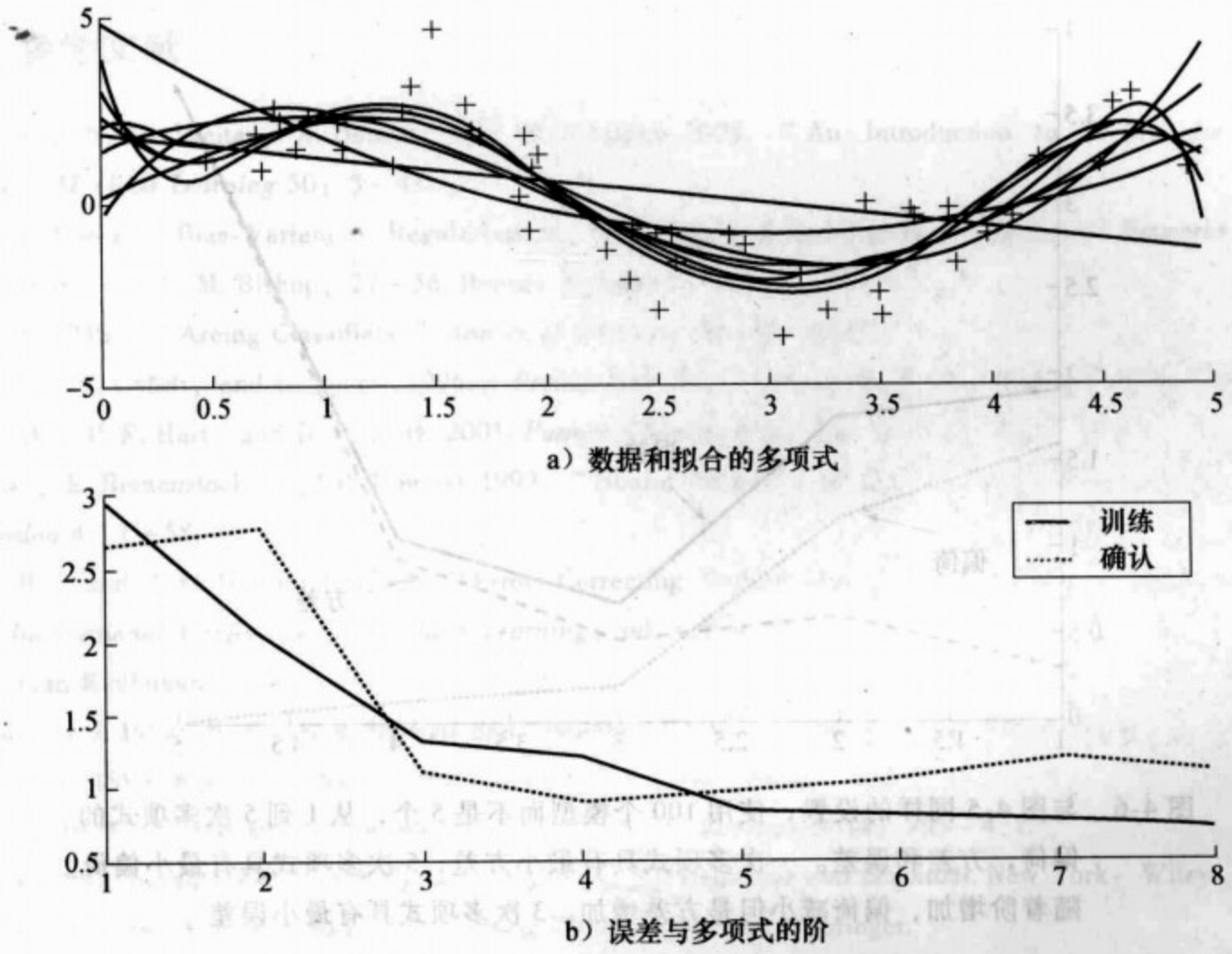


图 4-7 与图 4-5 同样的设置，产生训练集和确认集(每个包含 50 个实例)。a）训练数据和 1 到 8 阶拟合多项式。b）训练和确认误差作为多项式阶的函数。“拐点”在 3

另外一个常用的方法是正则化(regularization)(Breiman 1998a)。在这种方法中，我们用一个增广误差函数，记作

$$E' = \text{数据上的误差} + \lambda \cdot \text{模型复杂度} \tag{4.37}$$

它的第二项用一个大的方差惩罚复杂模型，其中 λ 给出了罚的权重。当我们最小化增广误差函数而不是数据上的误差时，我们惩罚了复杂模型，因此降低了方差。如果 λ 取得太大，则只允许很简单的模型，我们就会冒着引进偏倚的危险。使用交叉确认，λ 被优化。

77 ~ 80

结构风险最小化(structural risk minimization, SRM)(Vapnik 1995) 使用一个模型集，按它们复杂度排序。一个例子是阶递增的多项式。复杂度一般由自由参数的数量度量。VC 是另一种模型复杂度的度量。在(4.37)式中，我们可以使用递减的 λ_i 来得到复杂度递增的模型集合。SRM 模型选择对应于寻找最简单并且在数据上的经验误差最小的模型。

最小描述长度(minimum description length, MDL)(Rissanen 1978) 使用一种信息论度量。数据集的 *Kolmogorov* 复杂度定义为数据最短描述。如果数据简单，它就有短的复杂度；例如，如果它是 0 的序列，则我们可以只写 0 和序列的长度。如果数据完全随机的，则我们不能够有比数据自身更短的数据描述。如果一个模型对数据是合适的，则它有一个好的数据拟合，我们可以发送/存储数据描述而不是数据本身。在描述数据的所有模型中，我们想要有一个最简单的模型，那样它就可以有最短描述。这样，我们又一次要在模型的简单性和它解释数据能力之间权衡。

当我们有一些关于近似函数的合适类的先验知识时，我们使用贝叶斯模型选择(Bayes-

ian model selection)。这种先验知识被定义为模型的先验分布 p(模型)。给定数据并假定一个模型，我们可以用贝叶斯规则计算 p(模型 | 数据)：

$$p(\text{模型} \mid \text{数据}) = \frac{p(\text{数据} \mid \text{模型})p(\text{模型})}{p(\text{数据})} \tag{4.38}$$

p(模型 | 数据)是给定我们关于模型的主观先验知识(即 p(模型))和数据提供的客观支持(即 p(数据 | 模型))，模型的后验概率。我们可以选择具有最高后验概率的模型，或者用模型的后验概率加权，在所有模型上取平均。当选择先验使得最简单的模型具有最高概率时(根据奥克姆剃刀规则)，贝叶斯方法、正则化、SRM 和 MDL 是等价的。

交叉确认与其他模型选择的方法不同，因为它不对模型做任何先验假设。如果有足够大的确认数据集，它就是最好的方法。在数据样本很小时，其他模型变得有用。 81

4.9 注释

最大似然和贝叶斯估计基础的一个好资源是 Ross 1987。许多模式识别教材都讨论参数模型分类(例如，MacLachlan 1992；Devroye、Györfi 和 Lugosi 1996；Webb 1999；Duda、Hart 和 Stork 2001)。检查一元正态性的检验可以在 Rencher 1995 中找到。

Geman、Bienenstock 和 Doursat(1992)讨论了几个学习模型的偏倚和方差分解，这些我们也将在后边的章节讨论。偏倚/方差分解是针对回归的；对于分类，不同的研究者提出了偏倚和方差的不同定义；例如 Kong 和 Dietterich 1995 以及 Breiman 1998b。

4.10 习题

1. 写出产生以给定的 p 为参数的伯努利样本的程序，并且写出由样本计算 $\hat{p}$ 的程序。
2. 写出多项样本的对数似然，并证明(4.6)式。
3. 写出产生以给定 μ，σ 为参数的正态样本并由样本计算 m 和 s 的程序。对 μ 假定先验分布，用贝叶斯估计做相同的工作。
4. 给定两个正态分布 $p(x \mid C_1) \sim \mathcal{N}(\mu_1, \sigma_1^2)$ 和 $p(x \mid C_2) \sim \mathcal{N}(\mu_2, \sigma_2^2)$ 以及 $P(C_1)$ 和 $P(C_2)$，解析地计算贝叶斯判别点。
5. 在高斯密度的情况下，似然比 $\frac{p(x \mid C_1)}{p(x \mid C_2)}$ 是什么？
6. 对于两类问题，用不同的方差为两个类产生正态样本，然后使用参数分类法估计判别点。将它与理论值进行比较。
7. 假定一个线性模型，然后加入 0 均值的高斯噪声来产生一个样本。把样本一分为二，分别作为训练集和确认集。在训练集的这一半上使用线性回归。在确认集上计算误差。对 2 次和 3 次多项式进行同样的处理。
8. 当训练集较小时，方差对误差的贡献可能比偏倚大。在这种情况下，我们可能更喜欢简单模型，即使我们知道对于我们的任务它太简单。你能给出一个例子吗？ 82
9. 假设给定样本 $\mathcal{X}_i = \{x_i^t, r_i^t\}$，我们定义 $g_i(x) = r_i^1$，即我们对任意 x 的估计是数据集 $\mathcal{X}_i$ 的第一个实例(未排序)的 r 值。与 $g_i(x) = 2$ 和 $g_i(x) = \Sigma_t r_i^t / N$ 相比，关于它的偏倚和方差你有何评论？如果样本是有序的并使 $g_i(x) = \min_t r_i^t$，情况又如何？

4.11 参考文献

Andrieu, C., N. de Freitas, A. Doucet, and M. I. Jordan. 2003. "An Introduction to MCMC for Machine Learning." *Machine Learning* 50: 5 – 43.

Breiman, L. 1998a. "Bias- Variance, Regularization, Instability and Stabilization." In *Neural Networks and Machine Learning*, ed. C. M. Bishop, 27 – 56. Berlin: Springer.

Breiman, L. 1998b. "Arcing Classifiers." *Annals of Statistics* 26: 801 – 824.

Devroye, L., L. Györfi, and G. Lugosi. 1996. *A Probabilistic Theory of Pattern Recognition*. New York: Springer.

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, 2nd ed. New York: Wiley.

Geman, S., E. Bienenstock, and R. Doursat. 1992. "Neural Networks and the Bias/Variance Dilemma." *Neural Computation* 4: 1 – 58.

Kong, E. B., and T. G. Dietterich. 1995. "Error- Correcting Output Coding Corrects Bias and Variance." In *Twelfth International Conference on Machine Learning*, ed. A. Prieditis and S. J. Russell, 313 – 321. San Mateo, CA: Morgan Kaufmann.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Rencher, A. C. 1995. *Methods of Multivariate Analysis*. New York: Wiley.

Rissanen, J. 1978. "Modeling by Shortest Data Description." *Automatica* 14: 465 – 471.

Ross, S. M. 1987. *Introduction to Probability and Statistics for Engineers and Scientists*. New York: Wiley.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. New York: Springer.

Webb, A. 1999. *Statistical Pattern Recognition*. London: Arnold.

第5章 多元方法

在第4章，我们讨论了分类和回归的参数方法。现在，我们将它们推广到多元情况，其中我们有多个输入，并且输出(即类编码或连续输出)是这些输入的函数。这些输入可能是离散的或数值的。我们将讨论如何从标记的多元样本学习这样的函数，以及如何根据已有数据调整学习方法的复杂度。

5.1 多元数据

在许多应用中，许多测量都在每个个体或者事件上进行，并产生一个观测向量。样本可以看作一个数据矩阵(data matrix)

$$\mathbf{X} = \begin{bmatrix} X_1^1 & X_2^1 & \cdots & X_d^1 \\ X_1^2 & X_2^2 & \cdots & X_d^2 \\ \vdots & & & \\ X_1^N & X_2^N & \cdots & X_d^N \end{bmatrix}$$

其中 d 列对应 d 个变量，表示在个体或事件上的测量结果。它们也称为输入(input)、特征(feature)或属性(attribute)。N 行对应在 N 个个体或事件上的独立同分布的观测(observation)、样例(example)或实例(instance)。

例如，在对贷款申请做决定时，观测向量是与客户相关的一些信息，包括客户的年龄、婚姻状况、年收入等，并且我们有 N 个这样的老用户。这些测量也许有不同的尺度，例如，年龄用年计算，年收入用货币单位计算。某些(如年龄)可能是数值的，某些(如婚姻状况)可能是离散的。

通常，这些变量是相关的。如果它们不相关，就没有必要做多元分析。我们的目标也许是化简(simplification)，也就是用相对少的参数汇总大量数据。我们的目标也许是探测性的(exploratory)，并且我们可能对产生关于数据的假设感兴趣。在有些应用中，我们对由其他变量的值预测一个变量值很感兴趣。如果被预测变量是离散的，这就是多元分类问题；如果是数值的，这就是多元回归问题。

5.2 参数估计

均值向量(mean vector)$\boldsymbol{\mu}$ 的每个元素都是 $\mathbf{X}$ 一个列的均值：

$$E[\boldsymbol{x}] = \boldsymbol{\mu} = [\mu_1, \cdots, \mu_d]^T \tag{5.1}$$

X_i 的方差记作 σ_i^2，两个变量 X_i 和 X_j 的协方差定义为

$$\sigma_{ij} \equiv \mathrm{Cov}(X_i, X_j) = E[(X_i - \mu_i)(X_j - \mu_j)] = E[X_i X_j] - \mu_i \mu_j \tag{5.2}$$

满足 $\sigma_{ij} = \sigma_{ji}$，并且当 $i = j$ 时，$\sigma_{ii} = \sigma_i^2$。d 个变量就有 d 个方差和 $d(d-1)/2$ 个协方差。通常表示为 $d \times d$ 矩阵，称为协方差矩阵(covariance matrix)，用 $\boldsymbol{\Sigma}$ 表示，其第 (i, j) 个元素

是 σ_{ij}：

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & & & \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

对角线上的元素是方差，非对角线上的元素是协方差，并且矩阵是对称的。使用向量矩阵记号

$$\Sigma \equiv \text{Cov}(\boldsymbol{X}) = E[(\boldsymbol{X}-\boldsymbol{\mu})(\boldsymbol{X}-\boldsymbol{\mu})^T] = E[\boldsymbol{X}\boldsymbol{X}^T] - \boldsymbol{\mu}\boldsymbol{\mu}^T \tag{5.3}$$

86 如果两个向量是线性相关的，则协方差是正还是负取决于线性关系的斜率是正还是负。但是相关性的大小很难解释，因为它取决于两个变量的测量单位。变量 X_i 和 X_j 的相关性(correlation)是一个规范化到 -1 到 +1 之间的统计量，定义为

$$\text{Corr}(X_i, X_j) \equiv \rho_{ij} = \frac{\sigma_{ij}}{\sigma_i \sigma_j} \tag{5.4}$$

如果两个变量是相互独立的，那么其协方差为 0，因而相关性为 0。然而，其逆不真：变量也许是相关的(在非线性方式中)，然而它们的协方差可能为 0。

给定多元样本，可以计算这些参数的估计：均值的最大似然估计是样本均值 $\boldsymbol{m}$。它的第 i 维是 $\mathbf{X}$ 的第 i 列的平均值：

$$\boldsymbol{m} = \frac{\sum_{t=1}^N \boldsymbol{x}^t}{N}, \quad \text{其中 } m_i = \frac{\sum_{t=1}^N x_i^t}{N}, \quad i = 1, \cdots, d \tag{5.5}$$

Σ 的估计是样本协方差(sample covariance)矩阵 $\mathbf{S}$，其元素是

$$s_i^2 = \frac{\sum_{t=1}^N (x_i^t - m_i)^2}{N} \tag{5.6}$$

$$s_{ij} = \frac{\sum_{t=1}^N (x_i^t - m_i)(x_j^t - m_j)}{N} \tag{5.7}$$

它们是有偏估计，但如果在应用中估计的变化显著依赖于我们用 N 还是用 $N-1$ 来除，那么我们将遇到严重的麻烦。

样本相关(sample correlation)系数

$$r_{ij} = \frac{s_{ij}}{s_i s_j} \tag{5.8}$$

而样本的相关矩阵 $\mathbf{R}$ 包含 r_{ij}。

5.3 缺失值估计

观测中的某些变量的值可能缺失。最好的策略是把这些观测一同丢弃，但是，一般我们没有足够大的样本来让我们这样做，并且我们不想丢弃数据，因为非缺失的条目确实包含信

87 息。我们试图通过估计它们来填写缺失的条目，这被称作估算(imputation)。

在均值估算(mean imputation)中，对于数值变量，我们用现有数据的均值(平均值)来代替样本中缺失的变量值。对离散变量，我们用最可能出现的值，即数据中出现最多的值，来填写缺失的变量值。

在回归估算(imputation by regression)中，我们试图从其他已知变量的值来预测缺失的变量值。根据缺失变量的类型，我们分别定义一个回归或分类问题，用其值已知的数据点训练。如果许多不同变量都缺失，则我们取均值作为初始估计，并且反复执行该过程直到被预测的值稳定。如果这些变量不是高度相关的，则回归方法与均值估算等价。

然而，根据环境，有时特定属性值的缺失也许会很重要。例如，在信用卡申请中，如果申请人不提供他或她的电话号码，那也许是一条关键信息。在这样的情况下，我们用一个单独的值表示它，指明该值缺失并照此使用。

5.4 多元正态分布

在多元情况下，其中 $\boldsymbol{x}$ 是 d 维、正态分布的，我们有

$$p(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \tag{5.9}$$

并且我们记 $\boldsymbol{x}\sim\mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$，其中 $\boldsymbol{\mu}$ 是均值向量，$\boldsymbol{\Sigma}$ 是协方差矩阵(参见图 5-1)。正如

$$\frac{(x-\mu)^2}{\sigma^2} = (x-\mu)(\sigma^2)^{-1}(x-\mu)$$

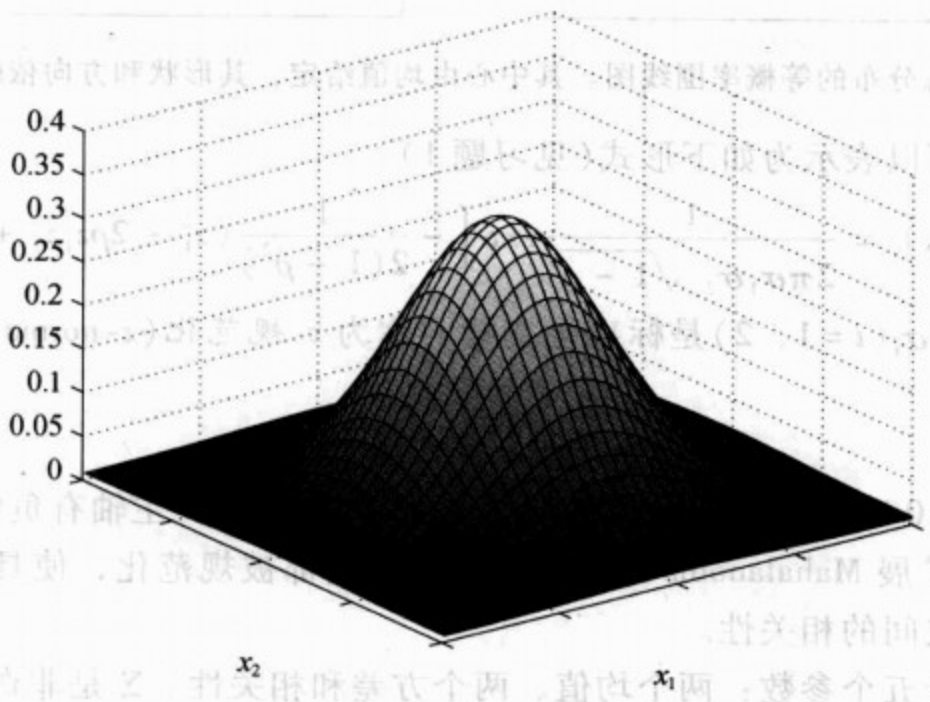


图 5-1 二元正态分布

是 x 到 μ 的以标准差为单位、对不同的方差规范化的平方距离一样，在多元情况下，使用 Mahalanobis 距离(Mahalanobis distance)：

$$(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \tag{5.10}$$

$(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = c^2$ 是以 $\boldsymbol{\mu}$ 为中心的 d 维超椭球，并且它的形状和方向由 $\boldsymbol{\Sigma}$ 决定。由于使用了 $\boldsymbol{\Sigma}$ 的逆，所以如果一个变量比其他变量的方差大，则它在 Mahalanobis 距离中的权重较小。类似地，两个高度相关变量的贡献没有两个相关性较低变量的贡献大。这样，使用协方差矩阵的逆具有将所有变量标准化(具有单位方差)并消除相关性的效果。

为便于显示，让我们考虑二元情况，其中 $d=2$(参见图 5-2)。当变量独立时，密度的主轴与输入轴平行。如果方差不同，则密度变成椭圆。密度根据协方差(相关性)的符号而旋转。均值向量为 $\boldsymbol{\mu}^T=[\mu_1, \mu_2]$，协方差矩阵通常表示为

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

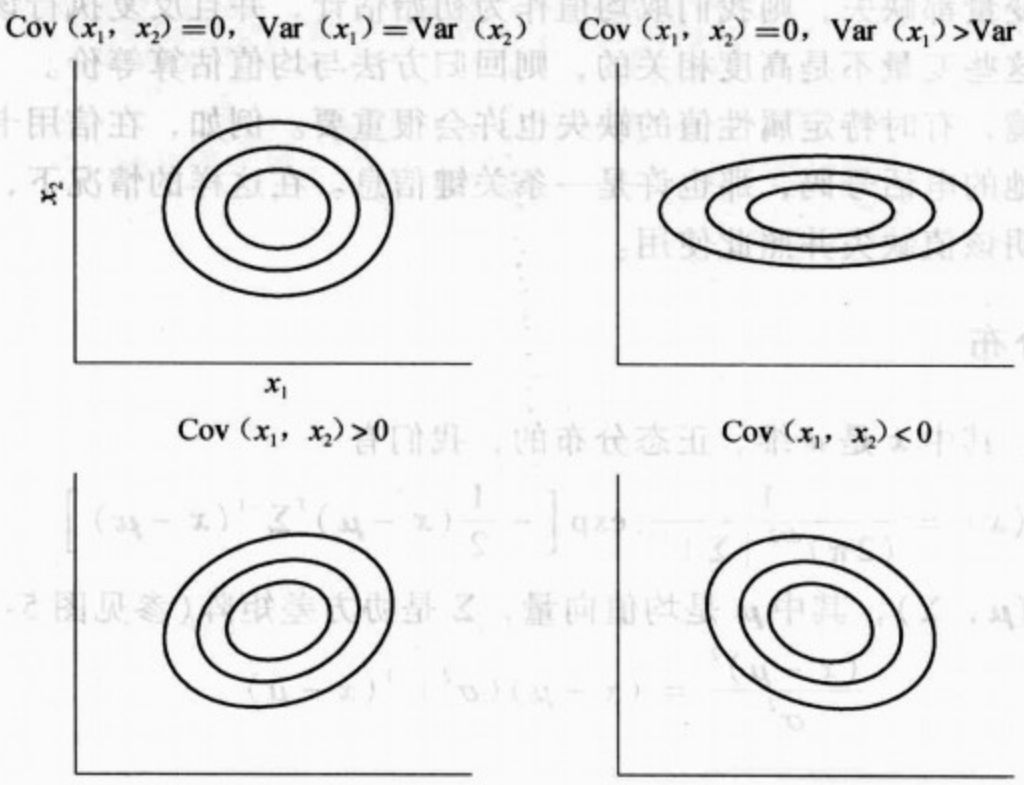


图 5-2 二元正态分布的等概率围线图。其中心由均值给定，其形状和方向依赖于协方差矩阵

二元联合密度可以表示为如下形式(见习题 1)

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}(z_1^2 - 2\rho z_1 z_2 + z_2^2)\right] \tag{5.11}$$

其中，$z_i = (x_i - \mu_i)/\sigma_i (i = 1, 2)$是标准化变量，称为 z-规范化(z-normalization)。记住，当 $|\rho| < 1$ 时，

$$z_1^2 + 2\rho z_1 z_2 + z_2^2 = \text{常数}$$

是椭圆方程。当 $\rho > 0$ 时，椭圆的主轴有正斜率，当 $\rho < 0$ 时，主轴有负斜率。

在(5.11)式的扩展 Mahalanobis 距离中，每个变量都被规范化，使具有单位方差，交叉项修正了两个变量之间的相关性。

概率密度依赖于五个参数：两个均值、两个方差和相关性。Σ 是非奇异的，因此是正定的，只要方差非 0 并且 $|\rho| < 1$。如果 ρ 是 +1 或者 −1，则两个变量是线性相关的，观测事实上是一维的，两个变量中的一个可以去掉。如果 $\rho = 0$，则两个变量是独立的，交叉项消失，我们得到两个一元密度的积。

88
~
90

在多元情况下，小的 $|\Sigma|$ 值表明样本接近 $\boldsymbol{\mu}$，正如单变量情况，小的 σ^2 表示样本接近 μ。小的 $|\Sigma|$ 还表示两个变量之间的高度相关性。Σ 是对称正定矩阵；这是 $\mathrm{Var}(X) > 0$ 的多元说法。否则，Σ 是奇异的，它的行列式的值为 0。这要么是由于维之间的线性依赖性，要么是因为有一维具有 0 方差。在这种情况下，应该将维度归约为正定矩阵；第 6 章将讨论这个问题的处理方法。

如果 $\boldsymbol{x} \sim \mathcal{N}_d(\boldsymbol{\mu}, \Sigma)$，则 $\boldsymbol{x}$ 的每个维都是一元正态的。(其逆不真：每一个 X_i 都可以是一元正态的，而 $\boldsymbol{X}$ 不一定是多元正态的。)实际上，变量的任意 $k < d$ 的子集都是 k 元正态的。一个特殊的情况是，$\boldsymbol{x}$ 的分量是独立的，并且 $i \neq j$ 时 $\mathrm{Cov}(X_i, X_j) = 0$，并且 $\mathrm{Var}(X_i) = \sigma_i^2$，

$\forall i$。于是，协方差矩阵是对角的，联合密度是各一元密度的乘积：

$$p(\boldsymbol{x}) = \prod_{i=1}^{d} p_i(x_i) = \frac{1}{(2\pi)^{d/2}\prod_{i=1}^{d}\sigma_i}\exp\left[-\frac{1}{2}\sum_{i=1}^{d}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2\right] \tag{5.12}$$

现在，我们考察另一个性质，它将在以后的章节中用到。我们假设 $\boldsymbol{x}\sim\mathcal{N}_d(\boldsymbol{\mu},\boldsymbol{\Sigma})$，其中 $\boldsymbol{w}\in\Re^d$，则

$$\boldsymbol{w}^T\boldsymbol{x} = w_1x_1 + w_2x_2 + \cdots + w_dx_d \sim \mathcal{N}\left(\boldsymbol{w}^T\boldsymbol{\mu}, \boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}\right)$$

给定

$$E[\boldsymbol{w}^T\boldsymbol{x}] = \boldsymbol{w}^TE[\boldsymbol{x}] = \boldsymbol{w}^T\boldsymbol{\mu} \tag{5.13}$$

$$\begin{aligned}\mathrm{Var}(\boldsymbol{w}^T\boldsymbol{x}) &= E[(\boldsymbol{w}^T\boldsymbol{x}-\boldsymbol{w}^T\boldsymbol{\mu})^2] = E[(\boldsymbol{w}^T\boldsymbol{x}-\boldsymbol{w}^T\boldsymbol{\mu})(\boldsymbol{w}^T\boldsymbol{x}-\boldsymbol{w}^T\boldsymbol{\mu})]\\ &= E[\boldsymbol{w}^T(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{w}] = \boldsymbol{w}^TE[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T]\boldsymbol{w}\\ &= \boldsymbol{w}^T\boldsymbol{\Sigma}\boldsymbol{w}\end{aligned} \tag{5.14}$$

这就是说，d 维正态分布在向量 $\boldsymbol{w}$ 上的投影是一元正态分布。在一般情况下，如果 $\mathbf{W}$ 是 $d\times k$ 矩阵，其秩 $k<d$，则 k 维 $\mathbf{W}^T\boldsymbol{x}$ 矩阵是 k 元正态分布：

$$\mathbf{W}^T\boldsymbol{x} \sim \mathcal{N}_k\left(\mathbf{W}^T\boldsymbol{\mu}, \mathbf{W}^T\boldsymbol{\Sigma}\mathbf{W}\right) \tag{5.15}$$

也就是说，如果我们把一个 d 维正态分布投影到 k 维的空间，则投影是 k 维正态分布。 91

5.5 多元分类

当 $\boldsymbol{x}\in\Re^d$ 时，如果取类条件密度 $p(\boldsymbol{x}\mid C_i)$ 为正态密度 $\mathcal{N}_d(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)$，则我们有

$$p(\boldsymbol{x}\mid C_i) = \frac{1}{(2\pi)^{d/2}\mid\boldsymbol{\Sigma}_i\mid^{1/2}}\exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right] \tag{5.16}$$

这样做的主要原因是它的分析简单性(Duda、Hart 和 Stork 2001)。此外，正态分布密度函数是许多自然现象的模型，因为大多数类的样本都可以看作是简单原型 $\boldsymbol{\mu}_i$ 的轻微改变版本，并且协方差矩阵 $\boldsymbol{\Sigma}_i$ 表示每个变量中的噪声量和这些噪声源的相关性。尽管真实数据可能并非常常是严格多元正态的，但是这是一个有用的近似。除了它易于进行数学处理外，该模型对偏离正态分布的鲁棒性在许多工作中都展示出来(例如 McLachlan 1992)。然而，一个明显的要求是一个类的样本应该形成单个组；如果有多个组，应该使用混合模型(第 7 章)。

假设我们要预测顾客可能感兴趣的汽车类型。不同的汽车是类，而 $\boldsymbol{x}$ 是顾客的可观测数据，例如年龄和收入。$\boldsymbol{\mu}_i$ 是购买 i 类汽车的顾客年龄和收入的均值向量，而 $\boldsymbol{\Sigma}_i$ 是它们的协方差矩阵：σ_{i1}^2 和 σ_{i2}^2 分别是年龄和收入的方差，并且 σ_{i12} 是购买 i 类汽车的顾客年龄和收入的协方差。

当我们定义判别式函数为

$$g_i(\boldsymbol{x}) = \log p(\boldsymbol{x}\mid C_i) + \log P(C_i)$$

并假定 $p(\boldsymbol{x}\mid C_i)\sim\mathcal{N}_d(\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i)$ 时，我们有

$$g_i(\boldsymbol{x}) = -\frac{d}{2}\log 2\pi - \frac{1}{2}\log\mid\boldsymbol{\Sigma}_i\mid - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i) + \log P(C_i) \tag{5.17}$$

给定 $K\geqslant 2$ 个类的训练样本 $\mathcal{X}=\{\boldsymbol{x}^t, \boldsymbol{r}^t\}$，其中如果 $\boldsymbol{x}^t\in C_i$，则 $r_i^t=1$，否则为 0。分别对每个类求最大似然，从而找到均值和协方差的估计：

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N} \tag{5.18}$$

$$\boldsymbol{m}_i = \frac{\sum_t r_i^t \boldsymbol{x}^t}{\sum_t r_i^t}$$

$$\mathbf{S}_i = \frac{\sum_t r_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T}{\sum_t r_i^t}$$

92

然后，将这些代入判别式函数，得到判别式的估计。忽略第一个常数项，我们有

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\log|\mathbf{S}_i| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_i)^T \mathbf{S}_i^{-1}(\boldsymbol{x} - \boldsymbol{m}_i) + \log \hat{P}(C_i) \tag{5.19}$$

把它展开，我们得到

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\log|\mathbf{S}_i| - \frac{1}{2}(\boldsymbol{x}^T \mathbf{S}_i^{-1}\boldsymbol{x} - 2\boldsymbol{x}^T \mathbf{S}_i^{-1}\boldsymbol{m}_i + \boldsymbol{m}_i^T \mathbf{S}_i^{-1}\boldsymbol{m}_i) + \log \hat{P}(C_i)$$

它定义了一个二次判别式(quadratic discriminant)(参见图 5-3)，也可以写作

$$g_i(\boldsymbol{x}) = \boldsymbol{x}^T \mathbf{W}_i \boldsymbol{x} + \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} \tag{5.20}$$

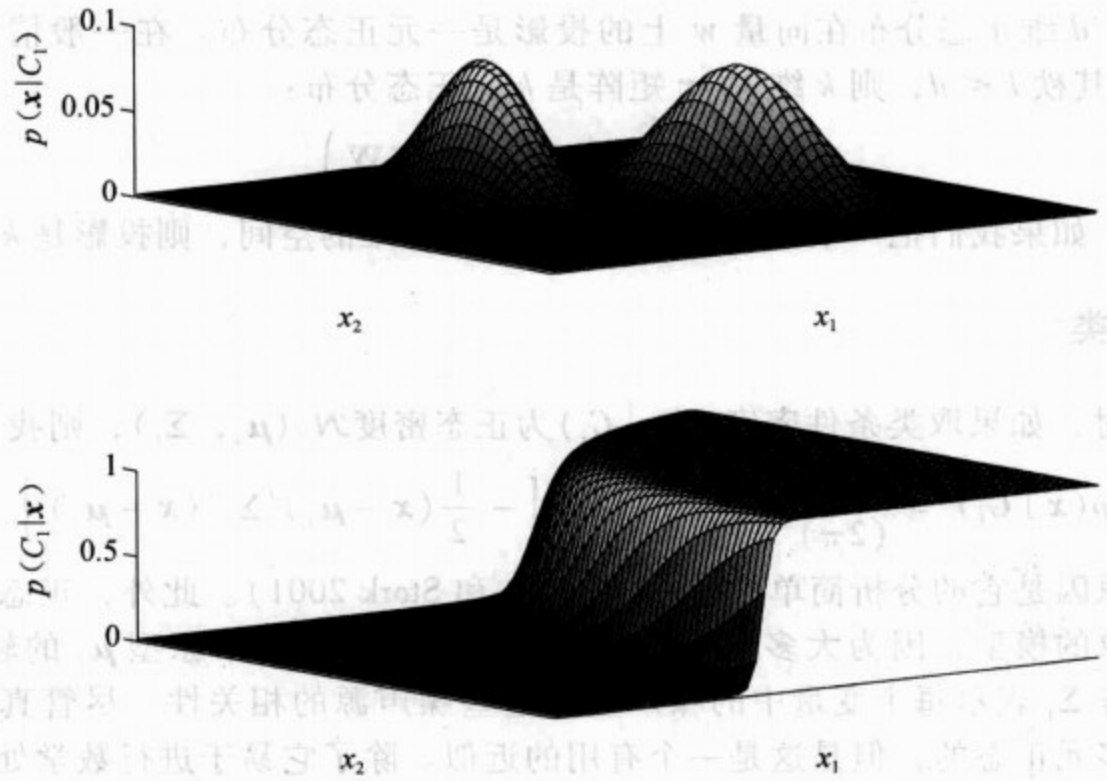


图 5-3 类具有不同的协方差矩阵。其中一个是类的似然密度和后验概率(顶部)。类分布用等概率围线表示，并且绘出判别式(底部)

其中

$$\mathbf{W}_i = -\frac{1}{2}\mathbf{S}_i^{-1}$$

$$\boldsymbol{w}_i = \mathbf{S}_i^{-1}\boldsymbol{m}_i$$

$$w_{i0} = -\frac{1}{2}\boldsymbol{m}_i^T\mathbf{S}_i^{-1}\boldsymbol{m}_i - \frac{1}{2}\log|\mathbf{S}_i| + \log\hat{P}(C_i)$$

对于均值，需要估计的参数数量为 $K \cdot d$ 个，而对于协方差矩阵为 $K \cdot d(d+1)/2$ 个。当 d 大而样本小时，$\mathbf{S}_i$ 可能是奇异的，并且其逆可能不存在。或者，$|\mathbf{S}_i|$ 可能非零但是太小，这种情况会不稳定；$\mathbf{S}_i$ 的小变化会引起 $\mathbf{S}_i^{-1}$ 的大变化。为了使小样本上的估计可靠，我们可能希望通过重新设计特性提取算法并选择特征子集，或者组合已有特征来降低维度 d。我们将在第 6 章讨论这样的方法。

另外一个可能的做法是汇集数据，并且对所有的类估计公共协方差矩阵：

$$\mathbf{S} = \Sigma_i\hat{P}(C_i)\mathbf{S}_i \tag{5.21}$$

在相同协方差矩阵的情况下，(5.19)式化简为

$$g_i(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m}_i)^T\mathbf{S}^{-1}(\boldsymbol{x}-\boldsymbol{m}_i) + \log\hat{P}(C_i) \tag{5.22}$$

对于均值，参数数量为 $K \cdot d$ 个，而对于共享协方差矩阵为 $d(d+1)/2$ 个。如果先验相等，则最佳决策规则是把输入指派到与输入均值的 Mahalanobis 距离最小的类。像前面一样，不相等的先验将边界移向不太可能的类。注意在这种情况下，二次项 $\boldsymbol{x}^T\mathbf{S}^{-1}\boldsymbol{x}$ 被取消，因为它出现在所有的判别式中，并且决策边界是线性的，导致如下线性判别式(linear discriminant)(图 5-4)

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} \tag{5.23}$$

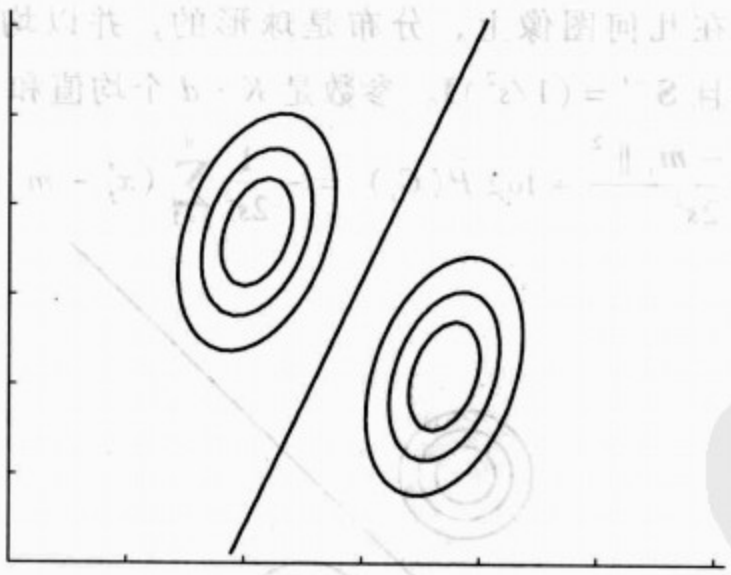


图 5-4 协方差可以是任意的，但是被两个类共享

其中

$$\boldsymbol{w}_i = \mathbf{S}^{-1}\boldsymbol{m}_i$$

$$w_{i0} = -\frac{1}{2}\boldsymbol{m}_i^T\mathbf{S}^{-1}\boldsymbol{m}_i + \log\hat{P}(C_i)$$

这种线性分类器的决策区域是凸的，即在一个决策区域内任意选择两个点并用一条直线连接，直线上的所有点都落在该区域内。

通过假定协方差矩阵的所有非对角线元素均为零，从而假定变量都是独立的，问题可以进一步简化。这是朴素贝叶斯分类(naive Bayes' classifier)，其中 $p(x_j \mid C_i)$ 是一元高斯的。$\mathbf{S}$ 和它的逆都是对角的，并且我们有

$$g_i(\boldsymbol{x}) = -\frac{1}{2}\sum_{j=1}^{d}\left(\frac{x_j^t - m_{ij}}{s_j}\right)^2 + \log \hat{P}(C_i) \tag{5.24}$$

93
ı
95

项 $((x_j^t - m_{ij})/s_j)^2$ 有规范化作用并以标准差为单位度量距离。从几何学角度来说，类是超椭圆体，并且因为协方差为零，它还是轴对齐的(见图5-5)。对于均值，参数的数量为 $K \cdot d$，而对于方差为 d。这样，$\mathbf{S}$ 的复杂度由 $\mathcal{O}(d^2)$ 降低为 $\mathcal{O}(d)$。

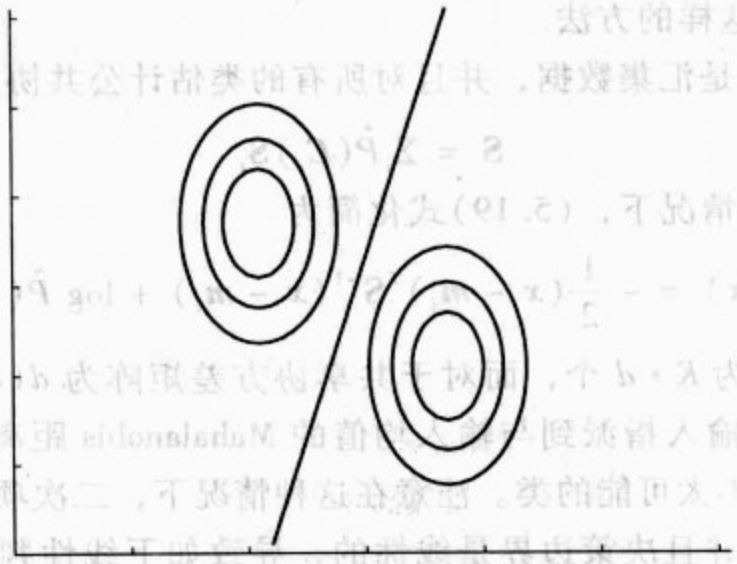


图 5-5 所有的类都具有相等的对角协方差，但是方差不相等

如果我们假定所有的变量是等同的，Mahalanobis 距离归结为欧氏距离(Euclidean distance)，问题还可以进一步简化。在几何图像上，分布是球形的，并以均值向量 $\boldsymbol{m}_i$ 为中心(见图5-6)。于是，$|\mathbf{S}| = s^{2d}$ 并且 $\mathbf{S}^{-1} = (1/s^2)\mathbf{I}$。参数是 $K \cdot d$ 个均值和一个 s^2。

$$g_i(\boldsymbol{x}) = -\frac{\|\boldsymbol{x} - \boldsymbol{m}_i\|^2}{2s^2} + \log \hat{P}(C_i) = -\frac{1}{2s^2}\sum_{j=1}^{d}(x_j^t - m_{ij})^2 + \log \hat{P}(C_i) \tag{5.25}$$

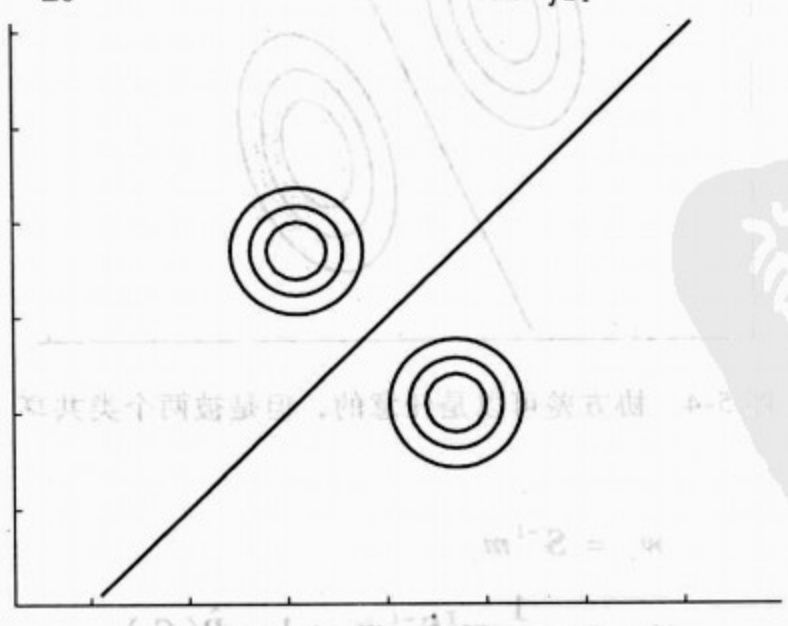


图 5-6 所有类具有相等的、在两个维上均具有相等方差的对角协方差矩阵

如果先验相等，则我们有 $g_i(\boldsymbol{x}) = -\|\boldsymbol{x}-\boldsymbol{m}_i\|^2$。这被称为最近均值分类(nearest mean classifier)，因为它把输入指派到最近均值的类。如果每个均值都被看作是类的理想原型或模板，那么这就是模板匹配(template matching)过程。它可以被扩展为

$$g_i(\boldsymbol{x}) = -\|\boldsymbol{x}-\boldsymbol{m}_i\|^2 = -(\boldsymbol{x}-\boldsymbol{m}_i)^T(\boldsymbol{x}-\boldsymbol{m}_i)$$
$$= -(\boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{m}_i^T\boldsymbol{x} + \boldsymbol{m}_i^T\boldsymbol{m}_i) \tag{5.26}$$

第一项 $\boldsymbol{x}^T\boldsymbol{x}$ 出现在所有的 $g_i(\boldsymbol{x})$ 中，可以去掉，并且我们可以把判别式函数写成

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0} \tag{5.27}$$

其中 $\boldsymbol{w}_i = \boldsymbol{m}_i$，$w_{i0} = -(1/2)\|\boldsymbol{m}_i\|^2$。如果所有的 $\boldsymbol{m}_i$ 有相似的范数，则 w_{i0} 也可以忽略，并且我们可以使用

$$g_i(\boldsymbol{x}) = \boldsymbol{m}_i^T\boldsymbol{x} \tag{5.28}$$

当 $\boldsymbol{m}_i$ 的范数可比较时，也可以使用点积代替(负的)欧氏距离作为相似性度量。

我们实际上可以把寻找最佳判别函数的任务看作是寻找最佳距离函数。这可以被看作是另外一种分类方法：我们不是要学习判别式函数 $g_i(\boldsymbol{x})$，而是要学习一个合适的距离函数 $\mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_2)$，使得对任意 $\boldsymbol{x}_1$，$\boldsymbol{x}_2$，$\boldsymbol{x}_3$，其中 $\boldsymbol{x}_1$，$\boldsymbol{x}_2$ 属于相同的类，而 $\boldsymbol{x}_1$，$\boldsymbol{x}_3$ 属于两个不同的类，我们希望有

$$\mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_2) < \mathcal{D}(\boldsymbol{x}_1, \boldsymbol{x}_3)$$

96 ~ 97

5.6 调整复杂度

在表 5-1 中，我们看到如何减少协方差矩阵的参数数目，在简单模型的适用性和通用性之间折衷。这是偏倚/方差两难选择的又一个例子。当我们做简化协方差矩阵的假设并降低被估计的参数数目时，我们就有引进偏倚的风险。另一方面，如果不做这种假设，并且矩阵是任意的，则二次判别式函数在小数据集上会有很大的方差。理想情况取决于已有数据所表示的问题的复杂度和我们所拥有的数据的规模。当我们拥有小数据集时，尽管协方差矩阵不同，但是假定共享协方差矩阵也许更好；单个协方差矩阵具有较少参数，并且可以利用更多的数据来估计，即用所有类的实例估计。这相当于使用线性判别式(linear discriminant)。分类经常用到线性判别式，我们将在第 10 章更详细地讨论它。

表 5-1 通过简化假设降低方差

假 设	协方差矩阵	参数数目
共享、超球	$\mathbf{S}_i = \mathbf{S} = s^2\mathbf{I}$	1
共享、轴对齐	$\mathbf{S}_i = \mathbf{S}$，其中 $s_{ij} = 0$	d
共享、超椭球	$\mathbf{S}_i = \mathbf{S}$	$d(d+1)/2$
不同、超椭球	$\mathbf{S}_i$	$K \cdot (d(d+1)/2)$

注意，当我们用欧氏距离度量相似性时，我们假设所有的变量都具有相同的方差，并且它们是相互独立的。在许多情况，这并不成立。例如，年龄与年收入具有不同的单位，并且在许多情况下是相关的。在这种情况下，可以在预处理阶段对输入进行 z-规范化(具

有0均值和单位方差），然后使用欧氏距离。另一方面，有些时候即使变量是相关的，如果我们没有足够的数据准确地计算相关性，也许最好还是假设它们不相关，并使用朴素贝叶斯分类。

Friedman(1989)提出一个组合所有这些特殊情况的方法，称为正则化判别式分析(regularized discriminant analysis，RDA)。我们还记得，正则化方法对应于从大方差开始到小方差的方法，有增加偏倚的风险。在利用高斯密度的参数化分类情况下，协方差矩阵可以表示成三种特殊情况的加权平均：

$$\mathbf{S}_i' = \alpha\sigma^2\mathbf{I} + \beta\mathbf{S} + (1-\alpha-\beta)\mathbf{S}_i \tag{5.29}$$

当 $\alpha=\beta=0$ 时，我们得到二次分类器。当 $\alpha=0$，$\beta=1$ 时，协方差矩阵被共享，我们得到线性分类器。当 $\alpha=1$，$\beta=0$ 时，协方差矩阵是对角阵，σ^2 在对角线上，我们得到最近均值分类。在这些极端情况之间，我们得到所有的不同分类方法，其中 α 和 β 通过交叉确认优化。

当数据集较小时，另一种正则化方法是通过定义 $\boldsymbol{\mu}_i$ 和 $\mathbf{S}_i$ 上的先验，使用贝叶斯方法，或者使用交叉确认选择表5-1中给出的四种情况中最好者。

5.7 离散特征

在许多应用中，我们有取 n 个不同值的离散属性。例如，一个属性可能是颜色∈{红，蓝，绿，黑}，另外一个可能是像素∈{有，无}。我们假设 x_j 是二元的(伯努利)，其中

$$p_{ij} \equiv p(x_j = 1 \mid C_i)$$

如果 x_j 是独立的二元变量，则我们有

$$p(\boldsymbol{x} \mid C_i) = \prod_{j=1}^{d} p_{ij}^{x_j}(1-p_{ij})^{(1-x_j)}$$

这是朴素贝叶斯分类的另一个例子，其中 $p(x_j \mid C_i)$ 是伯努利分布。判别式函数是

$$\begin{aligned} g_i(\boldsymbol{x}) &= \log p(\boldsymbol{x} \mid C_i) + \log P(C_i) \\ &= \sum_j [x_j \log p_{ij} + (1-x_j)\log(1-p_{ij})] + \log P(C_i) \end{aligned} \tag{5.30}$$

它是线性的。p_{ij} 的估计是

98
≀
99

$$\hat{p}_{ij} = \frac{\sum_t x_j^t r_i^t}{\sum_t r_i^t} \tag{5.31}$$

在一般情况下，我们假定 x_j 选自集合 $\{v_1, v_2, \cdots, v_{n_j}\}$。我们定义新的0/1哑变量

$$z_{jk}^t = \begin{cases} 1 & \text{如果 } x_j^t = v_k \\ 0 & \text{否则} \end{cases}$$

令 p_{ijk} 表示 x_j 属于类 C_i、取值为 v_k 的概率

$$p_{ijk} \equiv p(z_{jk} = 1 \mid C_i) = p(x_j = v_k \mid C_i)$$

如果属性是独立的，则我们有

$$p(\boldsymbol{x} \mid C_i) = \prod_{j=1}^{d}\prod_{k=1}^{n_j} p_{ijk}^{z_{jk}} \tag{5.32}$$

于是，判别式函数为

$$g_i(\boldsymbol{x}) = \sum_j \sum_k z_{jk} \log p_{ijk} + \log P(C_i) \tag{5.33}$$

p_{ijk}的最大似然估计为

$$\hat{p}_{ijk} = \frac{\sum_t z_{jk}^t r_i^t}{\sum_t r_i^t} \tag{5.34}$$

可以将它们代入(5.33)式中，得到判别式。

5.8 多元回归

在多元线性回归(multivariate linear regression)中，假定数值输出 r 为线性函数，即一些输入变量 x_1，…，x_d 和噪声的加权和。实际上，在统计学中，这称为多元(multiple)回归；当存在多个输出时，统计学家使用术语 multivariate(多元)。多元线性模型是

$$r^t = g(\boldsymbol{x}^t \mid w_0, w_1, \cdots, w_d) + \varepsilon = w_0 + w_1 x_1^t + w_2 x_2^t + \cdots + w_d x_d^t + \varepsilon \tag{5.35}$$

与一元情况相同，我们假设 ε 是正态的，具有 0 均值和常数方差。最大化该似然等价于最小化平方误差之和：

$$E(w_0, w_1, \cdots, w_d \mid \mathcal{X}) = \frac{1}{2} \sum_t (r^t - w_0 - w_1 x_1^t - w_2 x_2^t - \cdots - w_d x_d^t)^2 \tag{5.36}$$

100

关于参数 $w_j (j = 0, \cdots, d)$ 求导，我们得到正规方程(normal equation)：

$$\sum_t r^t = N w_0 + w_1 \sum_t x_1^t + w_2 \sum_t x_2^t + \cdots + w_d \sum_t x_d^t \tag{5.37}$$

$$\sum_t x_1^t r^t = w_0 \sum_t x_1^t + w_1 \sum_t (x_1^t)^2 + w_2 \sum_t x_1^t x_2^t + \cdots + w_d \sum_t x_1^t x_d^t$$

$$\sum_t x_2^t r^t = w_0 \sum_t x_2^t + w_1 \sum_t x_1^t x_2^t + w_2 \sum_t (x_2^t)^2 + \cdots + w_d \sum_t x_2^t x_d^t$$

$$\vdots$$

$$\sum_t x_d^t r^t = w_0 \sum_t x_d^t + w_1 \sum_t x_d^t x_1^t + w_2 \sum_t x_d^t x_2^t + \cdots + w_d \sum_t (x_d^t)^2$$

我们定义如下的向量与矩阵：

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots & x_d^1 \\ 1 & x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & & & & \\ 1 & x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \boldsymbol{r} = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix}$$

于是，正规方程可以写为

$$\mathbf{X}^T \mathbf{X} \boldsymbol{w} = \mathbf{X}^T \boldsymbol{r} \tag{5.38}$$

并且我们可以求解参数

$$\boldsymbol{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \boldsymbol{r} \tag{5.39}$$

这个方法与我们在单个输入的多项式回归中所使用的方法一样。如果我们定义变量为 $x_1 = x$，$x_2 = x^2$，…，$x_k = x^k$，则两个问题是一样的。这也提示我们，如果需要的话，如何做多元多项式回归(multivariate polynomial regression)(参见习题 5)。但是，除非 d 很小，否则

在多元回归中，我们很少使用比线性更高阶的多项式。线性模型的一个优点是，回归之后，观察 $w_j(j=1, \cdots, d)$ 的值，我们可以提取知识：首先，观察 w_j 的符号，我们就知道 x_j 对输出结果的影响是正的还是负的。第二，如果所有的 x_j 具有相同的值域，则通过观测 w_j 的绝对值，我们可以知道特征的重要性，并按重要性为特征定秩，甚至可以去掉那些 w_j 接近于0的特征。

101 当有多个输出时，可以等价地定义一组独立的单输出回归问题。

5.9 注释

一本更新我们的线性代数知识的好书是 Strang 1988。Harville 1997 是另外一本很好的书，它从统计学的角度处理矩阵代数。

用多元变量的一个不便之处是，当维数很大时，不能够进行可视分析。统计学文献中已经提出了一些方法，来显示多元数据；Rencher 1995 给出了综述。一种可能的方法是两两变量绘制二元散点图：如果数据是多元正态的，则任意两个变量的图应该是大致上线性的；这可以用作多元正态性的视觉检测。我们将在第6章中讨论的另外一个可能的方法是把它们投影到一维或两维上，并在那里显示。

模式识别的大部分工作都是在假定多元正态密度上进行的。有时，这样的判别式甚至被称为贝叶斯最优分类，但是这通常是错误的；只有当密度确实是多元正态的，并且我们有足够的数据来计算正确的参数时，它才是最优的。Rencher 1995 讨论了评估多元正态性的检验，以及检查相等协方差矩阵的检验。McLachlan 1992 讨论了用多元正态分布分类，并且比较了线性和二次判别式。

多元正态分布的一个明显的约束是它不允许某些特征是离散型数据。一个具有 n 个可能值的变量可以被转化成 n 个 0/1 哑变量，但是这增加了维度。我们可以用第6章中介绍的方法在这个 n 维空间上进行维度归约，从而不会增加维度。对于这种混合特征的参数分类，McLachlan 1992 有详细的讨论。

5.10 习题

1. 证明(5.11)式。
2. 从多元正态密度 $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ 产生一个样本，计算 $\boldsymbol{m}$ 和 $\mathbf{S}$ 并将它们与 $\boldsymbol{\mu}$ 和 Σ 比较。检查样本大小变化时估计的变化情况。
3. 从两个多元正态密度 $\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)(i=1, 2)$ 产生样本，并对表5-1中的四种情况计算贝叶斯最优判别式。

102

4. 对于两类问题，针对表5-1中高斯密度的四种情况，推导：

$$\log \frac{P(C_1 \mid \boldsymbol{x})}{P(C_2 \mid \boldsymbol{x})}$$

5. 假设我们有两个变量 x_1 和 x_2，并且我们想对它们做二次拟合，即：

$$f(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 (x_1)^2 + w_5 (x_2)^2$$

给定样本$\mathcal{X}=\{x_1^t, x_2^t, r^t\}$，如何找到$w_i(i=0, \cdots, 5)$？

5.11 参考文献

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, 2nd ed. New York: Wiley.

Friedman, J. H. 1989. "Regularized Discriminant Analysis." *Journal of American Statistical Association* 84: 165 – 175.

Harville, D. A. 1997. *Matrix Algebra from a Statistician's Perspective*. New York: Springer.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Rencher, A. C. 1995. *Methods of Multivariate Analysis*. New York: Wiley.

Strang, G. 1988. *Linear Algebra and its Applications*, 3rd ed. New York: Harcourt Brace Jovanovich.

第6章 维度归约

任何分类和回归方法都依赖于输入的数量。这决定了时间和空间的复杂度以及训练这样的分类器和回归器所需要的训练样例数量。本章中，我们讨论各种降低输入维度而不损失准确率的方法。

6.1 引言

在一个应用中，无论它是分类还是回归，我们确信含有信息的观测数据都被用作输入并且输入到系统中来做决策。理想情况下，我们不应该将特征选择或特征提取作为一个单独的进程；分类方法(或回归方法)应该能够利用任何必要的特征，而丢弃不相关的特征。然而，有许多原因使得我们对把降维作为一个单独的预处理步骤感兴趣：

- 在大多数学习算法中，复杂度依赖于输入的维度 d 和数据样本的规模 N，并且为了减少存储量和计算时间，我们对降低问题的维度感兴趣。降低 d 也降低了检验时推理算法的复杂度。
- 当一个输入被认定并不必要时，我们就节省了提取它的开销。
- 较简单的模型在小数据集上更为鲁棒。较简单的模型具有较小的方差，也就是说，它们的变化更少地依赖于样本的特殊性，包括噪声、离群点等。
- 当数据能够用较少特征解释时，我们就能够更好地理解解释数据的过程，这使得我们能够提取知识。
- 当数据可以用少数维表示而不丢失信息时，我们可以对数据绘图，并可视化地分析它的结构和离群点。

降低维度的主要方法有两种：特征选择和特征提取。在特征选择(feature selection)中，我们感兴趣的是从 d 维中找出为我们提供最多信息的 k 个维，并且丢弃其他的 $(d-k)$ 个维。作为一种特征选择方法，我们将要讨论子集选择(subset selection)。

在特征提取(feature extraction)中，我们感兴趣的是找出 k 个维的新集合，这些维是原来 d 个维的组合。这些方法可以是监督的，也可以是非监督的，这取决于它们是否使用输出信息。最著名和最广泛使用的特征提取方法是主成分分析(PCA)和线性判别分析(LDA)。它们都是线性投影方法，分别是非监督和监督的。PCA 与其他两种非监督的线性投影方法有许多相似之处。我们也将讨论这两种方法——因子分析(FA)和多维定标(MDS)。

6.2 子集选择

在子集选择(subset selection)中，我们对发现特征集中的最佳子集感兴趣。最佳子集包含的维最少，而它们对正确率的贡献最大。我们丢弃剩余的不重要的维。使用一个合适的误差函数，最佳子集在回归和分类问题中都可以使用。d 个变量有 2^d 个可能子集，但是除非 d

很小，否则我们不能对所有子集进行检验。我们使用启发式的方法，在合理的(多项式)时间内得到一个合理的(但不是最优的)解。

有两种方法：在向前选择(forward selection)中，我们从空集开始，逐个添加它们，每次添加一个降低误差最多的变量，直到进一步的添加不会降低误差(或降低很少)。在向后选择(backward selection)中，我们从所有变量开始，逐个排除它们，每次排除一个降低误差最 106
多(或提高很少)的变量，直到进一步的排除会显著提高误差。在这两种情况下，误差检测都应该在不同于训练集的确认集上进行，因为我们想要检验泛化准确率。使用更多的特征，我们一般会有更低的训练误差，但是不一定有更低的确认误差。

我们用 F 表示输入维的特征 $x_i(i=1, \cdots, d)$ 的集合，$E(F)$ 表示当我们只使用 F 中的输入时，在确认样本上出现的误差。依赖于应用，误差或者是均方差误差，或者是误分类错误。

在顺序向前选择(sequential forward selection)中，我们从 $F=\varnothing$ 开始。每一步中，我们针对所有可能的 x_i，训练我们的模型并在确认集上计算 $E(F \cup x_i)$。然后，我们选择导致最小误差的输入 x_j

$$j = \arg\min_i E(F \cup x_i) \tag{6.1}$$

并且我们

$$\text{将 } x_j \text{ 添加到 } F \text{ 中，如果 } E(F \cup x_j) < E(F) \tag{6.2}$$

如果添加任何特征都不会减少 E，则我们停止。如果误差降低太小，我们甚至可以决定提前停止；这里存在一个用户定义的阈值，依赖于应用约束以及错误和复杂度的折中。增加另外一个特征带来观测该特征的开销，也会使分类器/回归器更加复杂。

这样的过程也许开销很大，因为将 d 维减少到 k 维，我们需要训练和测试系统 $d+(d-1)+(d-2)+\cdots+(d-k)$ 次，其复杂度为 $O(d^2)$。这是一个局部搜索过程，并且不能保证找到最佳子集，即导致最小误差的最小子集。例如，x_i 和 x_j 本身可能不好，但是合起来却可能会把误差降低很多。但是该算法很贪婪，逐个增加特征，因此它也许不能发现 x_i 与 x_j 的并。以更多计算为代价，一次增加 m 个而不是一个特征是可能的。我们还可以在当前添加之后回溯并且检查以前添加的哪个特征可以去掉，这增大了搜索空间但是也增加了复杂度。在浮动搜索(floating search)方法中(Pudil, Novovičová 和 Kittler 1994)，每一步还可以改变增加和去掉的特征数量。 107

在顺序向后选择(sequential backward selection)中，我们从包括所有特征的 F 开始，并且执行类似的过程，但是与添加相反，我们从 F 中去掉一个特征，并且是去掉导致误差最小的那个

$$j = \arg\min_i E(F - x_i) \tag{6.3}$$

而我们

$$\text{从 } F \text{ 中去掉 } x_j \text{，如果 } E(F - x_j) < E(F) \tag{6.4}$$

如果去掉特征不能降低误差时我们就停止。为了降低复杂度，我们可能也会决定去掉一个特征，如果它的去掉只引起很轻微的误差增加。

向前搜索的所有可能变体对于向后搜索也是可行的。向后搜索与向前搜索具有相同的复杂度。但是，训练具有较多特征的系统比较训练具有较少特征的系统开销更大，并且如果我们预料有许多无用特征时，向前搜索更可取。

子集选择是监督的，因为输出被回归器或分类器用作计算误差，但是它可以用于任何回

归和分类方法。在多元正态分类的特殊情况下，如果原来的 d 维类密度是多元正态的，则其任意子集也是多元正态的，并且仍然可以使用参数分类，并具有用 $k \times k$ 维协方差矩阵代替 $d \times d$ 维协方差矩阵的优点。

在像人脸识别这样的应用中，特征选择不是降维的好方法，因为个体像素本身并不携带很多识别信息；携带脸部识别信息的是许多像素值的组合。这可以通过我们后边将要讨论的特征提取方法来做。

6.3 主成分分析

108 在投影方法中，我们感兴趣的是找到一个从原 d 维输入空间到新的($k < d$)维空间的、具有最小信息损失的映射。$\boldsymbol{x}$ 在方向 $\boldsymbol{w}$ 上的投影为

$$z = \boldsymbol{w}^T \boldsymbol{x} \tag{6.5}$$

主成分分析(principal components analysis, PCA)是一种非监督方法，因为它不使用输出信息；需要最大化的是方差。主成分是这样的 $\boldsymbol{w}_1$，样本投影到 $\boldsymbol{w}_1$ 上之后被广泛散布，使得样本点之间的差别变得最明显。为了得到唯一解并且使该方向成为最重要因素，我们要求 $\| \boldsymbol{w}_1 \| = 1$。从(5.14)式我们知道，如果 $z_1 = \boldsymbol{w}^T \boldsymbol{x}$，并且 $\mathrm{Cov}(\boldsymbol{x}) = \Sigma$，则

$$\mathrm{Var}(z_1) = \boldsymbol{w}_1^T \Sigma \boldsymbol{w}_1$$

我们寻找 $\boldsymbol{w}_1$，使得 $\mathrm{Var}(z_1)$受限于约束 $\boldsymbol{w}_1^T \boldsymbol{w}_1 = 1$ 最大化。将这写成拉格朗日问题，我们得到

$$\max_{\boldsymbol{w}_1} \boldsymbol{w}_1^T \Sigma \boldsymbol{w}_1 - \alpha(\boldsymbol{w}_1^T \boldsymbol{w}_1 - 1) \tag{6.6}$$

关于 $\boldsymbol{w}_1$ 求导并令它等于0，我们有

$$2\Sigma \boldsymbol{w}_1 - 2\alpha \boldsymbol{w}_1 = 0, \quad \text{因此 } \Sigma \boldsymbol{w}_1 = \alpha \boldsymbol{w}_1$$

如果 $\boldsymbol{w}_1$ 是 Σ 的本征向量，α 是对应的本征值，则上式成立。因为我们有

$$\boldsymbol{w}_1^T \Sigma \boldsymbol{w}_1 = \alpha \boldsymbol{w}_1^T \boldsymbol{w}_1 = \alpha$$

为了方差最大，我们选择具有最大本征值的本征向量。因此，主成分是输入样本的协方差矩阵的具有最大本征值 $\lambda_1 = \alpha$ 的本征向量。

第二个主成分 $\boldsymbol{w}_2$ 也应该最大化方差，具有单位长度，并且与 $\boldsymbol{w}_1$ 正交。后者的要求是使得投影后 $z_2 = \boldsymbol{w}_2^T \boldsymbol{x}$ 与 z_1 不相关。对于第二个主成分，我们有

$$\max_{\boldsymbol{w}_2} \boldsymbol{w}_2^T \Sigma \boldsymbol{w}_2 - \alpha(\boldsymbol{w}_2^T \boldsymbol{w}_2 - 1) - \beta(\boldsymbol{w}_2^T \boldsymbol{w}_1 - 0) \tag{6.7}$$

关于 $\boldsymbol{w}_2$ 求导并令它等于0，我们有

109
$$2\Sigma \boldsymbol{w}_2 - 2\alpha \boldsymbol{w}_2 - \beta \boldsymbol{w}_1 = 0 \tag{6.8}$$

用 $\boldsymbol{w}_1^T$ 左乘，我们得到

$$2\boldsymbol{w}_1^T \Sigma \boldsymbol{w}_2 - 2\alpha \boldsymbol{w}_1^T \boldsymbol{w}_2 - \beta \boldsymbol{w}_1^T \boldsymbol{w}_1 = 0$$

注意 $\boldsymbol{w}_1^T \boldsymbol{w}_2 = 0$。$\boldsymbol{w}_1^T \Sigma \boldsymbol{w}_2$ 是标量，等于它的转置 $\boldsymbol{w}_2^T \Sigma \boldsymbol{w}_1$，这里 $\boldsymbol{w}_1$ 是 Σ 的主本征向量，$\Sigma \boldsymbol{w}_1 = \lambda_1 \boldsymbol{w}_1$，因此

$$\boldsymbol{w}_1^T \Sigma \boldsymbol{w}_2 = \boldsymbol{w}_2^T \Sigma \boldsymbol{w}_1 = \lambda_1 \boldsymbol{w}_2^T \boldsymbol{w}_1 = 0$$

于是 $\beta = 0$，并且(6.8)式可以简化为

$$\Sigma \boldsymbol{w}_2 = \alpha \boldsymbol{w}_2$$

这表明 $\boldsymbol{w}_2$ 应该是 Σ 的本征向量，具有第二大本征值 $\lambda_2 = \alpha$。类似地，我们可以证明其他维

被具有递减的本征值的本征向量给出。

因为 Σ 是对称的，因此对于两个不同的本征值，本征向量是正交的。如果 Σ 是正定的(对于所有的非空 $\boldsymbol{x}$，$\boldsymbol{x}^T\Sigma\boldsymbol{x}>0$)，则它的所有本征值都是正的。如果 Σ 是奇异的，则它的秩(有效维数)为 k，并且 $k<d$，$\lambda_i(i=k+1, \cdots, d)$均为0($\lambda_i$ 以递减序排序)。k 个具有非零本征值的本征向量是约化空间的维。第一个本征向量(具有最大本征值的向量)$\boldsymbol{w}_1$(即为主成分)贡献了方差的最大部分，第二个贡献了方差的第二大部分，依此类推。

我们定义

$$\boldsymbol{z} = \mathbf{W}^T(\boldsymbol{x} - \boldsymbol{m}) \tag{6.9}$$

其中 $\mathbf{W}$ 的 k 列是 $\mathbf{S}$ 的 k 个主本征向量，也是 Σ 的估计。我们在 $\boldsymbol{x}$ 投影前减去均值 $\boldsymbol{m}$，将数据在原点中心化。该线性变换后，我们得到 k 维空间，它的维是本征向量，并且在这些新维上的方差等于本征值(见图 6-1)。为了规范化方差，我们可以除以本征值的平方根。

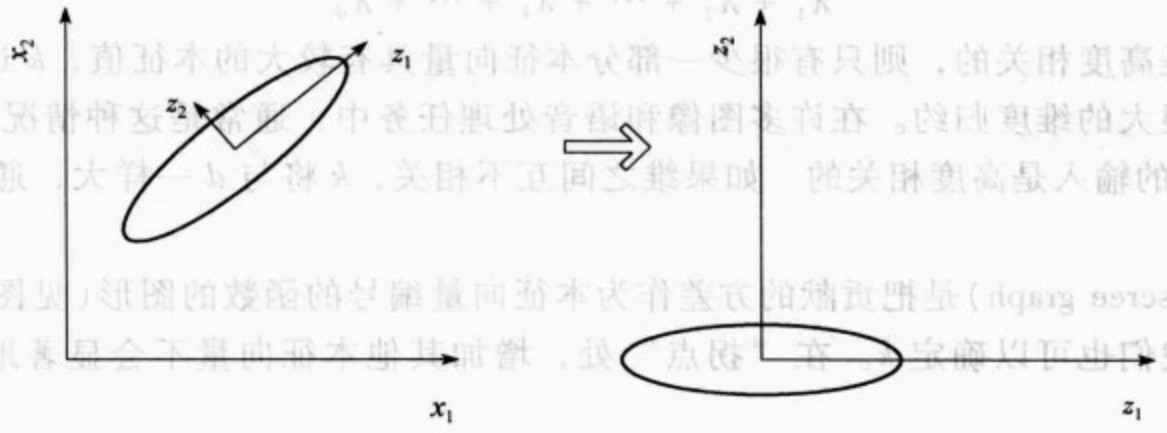


图 6-1 主成分分析使样本中心化，然后旋转坐标轴与最大方差方向一致。如果 z_2 上的方差太小，则可以忽略它，并且我们得到从二维到一维的维度归约

让我们来看另一种推导：我们想要找到一个矩阵 $\mathbf{W}$，使得当我们有 $\boldsymbol{z}=\mathbf{W}^T\boldsymbol{x}$(假设不失一般性，$\boldsymbol{x}$ 已经被中心化)时，我们将得到 $\mathrm{Cov}(\boldsymbol{z})=\mathbf{D}'$，其中 $\mathbf{D}'$是任意对角阵；也就是说，我们希望得到不相关的 z_i。

如果我们建立一个 $d\times d$ 的矩阵 $\mathbf{C}$，其第 i 列是 $\mathbf{S}$ 的规范化的本征向量 $\boldsymbol{c}_i$，则 $\mathbf{C}^T\mathbf{C}=\mathbf{I}$，并且

$$\begin{aligned}
\mathbf{S} &= \mathbf{S}\mathbf{C}\mathbf{C}^T \\
&= \mathbf{S}(\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_d)\mathbf{C}^T \\
&= (\mathbf{S}\boldsymbol{c}_1, \mathbf{S}\boldsymbol{c}_2, \cdots, \mathbf{S}\boldsymbol{c}_d)\mathbf{C}^T \\
&= (\lambda_1\boldsymbol{c}_1, \lambda_2\boldsymbol{c}_2, \cdots, \lambda_d\boldsymbol{c}_d)\mathbf{C}^T \\
&= \lambda_1\mathbf{c}_1\boldsymbol{c}_1^T + \cdots + \lambda_d\boldsymbol{c}_d\boldsymbol{c}_d^T \\
&= \mathbf{C}\mathbf{D}\mathbf{C}^T
\end{aligned} \tag{6.10}$$

其中 $\mathbf{D}$ 是对角矩阵，其对角线元素是本征值 λ_1，…，λ_d。这称为 $\mathbf{S}$ 的谱分解(spectral decomposition)。由于 $\mathbf{C}$ 是正交的，并且 $\mathbf{C}\mathbf{C}^T=\mathbf{C}^T\mathbf{C}=\mathbf{I}$，我们可以在上式左乘以 $\mathbf{C}^T$，右乘以 $\mathbf{C}$，得到

$$\mathbf{C}^T\mathbf{S}\mathbf{C} = \mathbf{D} \tag{6.11}$$

我们知道如果 $\boldsymbol{z}=\mathbf{W}^T\boldsymbol{x}$，则 $\mathrm{Cov}(\boldsymbol{z})=\mathbf{W}^T\mathbf{S}\mathbf{W}$，我们希望它等于一个对角矩阵。于是，从(6.11)式我们看到，可以令 $\mathbf{W}=\mathbf{C}$。

110
~
111

让我们看一个例子，以便得到一些直观体验(Rencher 1995)：假设我们有一班学生的五门课程的成绩，并且我们希望对这些学生排序。也就是说，我们希望把这些数据投影到一个维上，使得这些数据点之间的差别最明显。我们可以用PCA。具有最大本征值的本征向量是最大方差的方向，也就是学生最为分散的方向。这样做比计算平均值好，因为我们考虑了方差的相关性和区别。

实践中，即使所有本征值都大于0，但是如果$|\mathbf{S}|$很小(注意$|\mathbf{S}| = \prod_{i=1}^{d} \lambda_i$)，那么我们知道，某些本征值对方差影响很小，并且可以丢弃。因此，我们考虑例如贡献90%以上方差的前k个主要成分。当λ_i降序排列时，由前k个主要成分贡献的*方差比例*(proportion of variance)为

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

如果维是高度相关的，则只有很少一部分本征向量具有较大的本征值，k远比d小，并且可能得到很大的维度归约。在许多图像和语音处理任务中，通常是这种情况，其中(时间或空间)邻近的输入是高度相关的。如果维之间互不相关，k将与d一样大，通过PCA就没有收益。

斜坡图(scree graph)是把贡献的方差作为本征向量编号的函数的图形(见图6-2)。通过目视分析，我们也可以确定k。在“拐点”处，增加其他本征向量不会显著地增加贡献的方差。

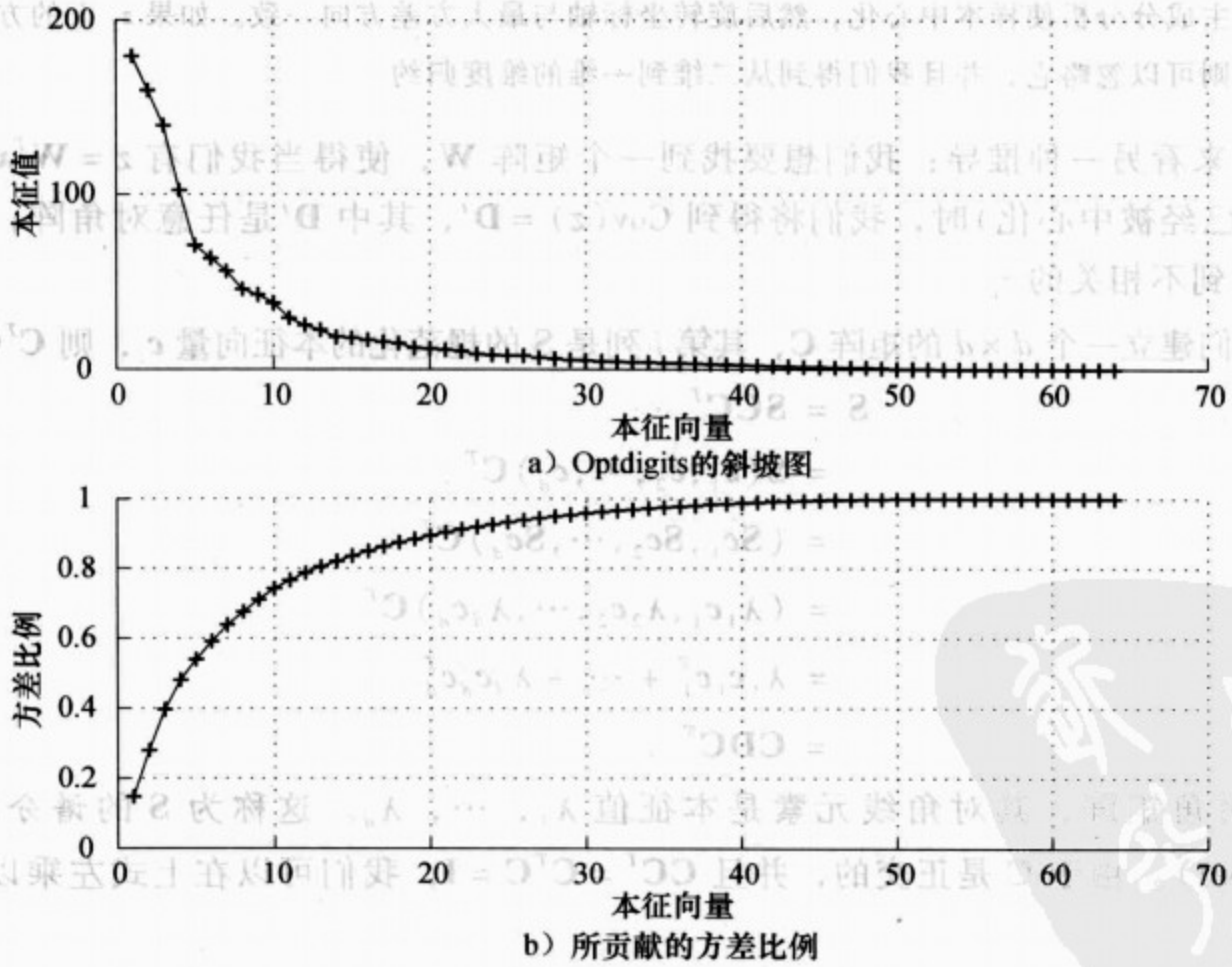


图6-2 a)斜坡图。b)对于取自UCI知识库的Optdigits数据集，显示所贡献的方差比例。Optdigits是手写数字数据集，具有10个类和64维输入。前20个本征向量贡献了90%的方差

另一个可能的方法是忽略那些本征值小于平均输入方差的本征向量。给定 $\Sigma_i \lambda_i = \Sigma_i s_i^2$（等于矩阵 **S** 的迹，记作 tr(**S**)），平均本征值等于平均输入方差。当我们仅保留本征值大于平均本征值的本征向量时，我们仅保留了那些其方差大于平均输入方差的本征向量。

如果原 x_i 维的方差变化显著，则它们对主成分方向的影响比相关性大。因此，一个公共过程是在使用 PCA 之前对数据进行预处理，使得每个维都具有 0 均值和单位方差。或者，为了使协方差而不是个体方差起作用，我们可以使用协相关矩阵 **R** 而不是协方差矩阵 **S** 的本征向量。

PCA 解释方差并对离群点很敏感：少量远离中心的点对方差有很大影响，从而也对本征向量有很大影响。鲁棒的估计（Robust estimation）方法允许计算离群点存在时的参数。一种简单的方法是计算数据点的 Mahalanobis 距离，丢弃那些远离的孤立数据点。

如果前两个主成分贡献方差的很大百分比，则我们可以做目视分析：我们可以在这个二维空间绘制数据（见图 6-3），目视地搜索结构、组、离群点、正态性等等。相对于原来的任何两个变量的图，该图对样本给出了更好的图形描述。通过观察主成分的维，我们还可以试着揭示一些有意义的描述数据的潜在变量。例如，在图像应用方面，输入是图像，本征向量可以显示为图像，并且可以看作重要特征的模板；它们常常被形象地称为“本征面孔”（eigenface）、“本征数字”（eigendigit）等（Turk 和 Pentland 1991）。

112 ~ 113

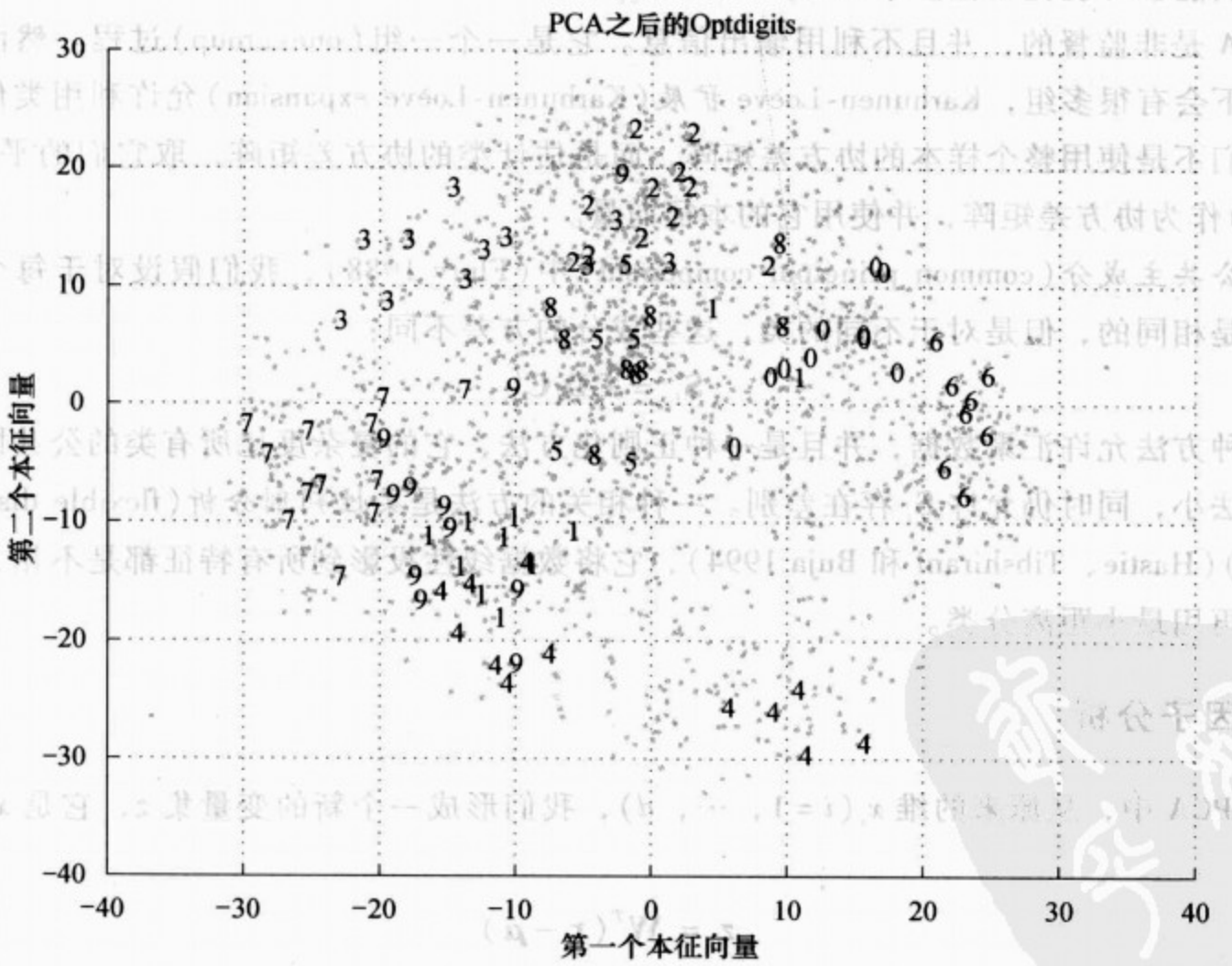


图 6-3　绘制在两个主成分空间的 Optdigits 数据。只显示了 100 个数据点的标号，以便最小化墨噪比（ink-to-noise ratio）

当 d 很大时，计算、存储、处理 $\mathbf{S}$ 都很繁琐。我们可以直接从数据计算本征向量、本征值，而不必显式地计算协方差矩阵(Chatfield 和 Collins 1980)。

从(5.15)式我们知道，如果 $\boldsymbol{x} \sim \mathcal{N}_d(\boldsymbol{\mu}, \boldsymbol{\Sigma})$，则投影后 $\mathbf{W}^T\boldsymbol{x} \sim \mathcal{N}_k\left(\mathbf{W}^T\boldsymbol{\mu}, \mathbf{W}^T\boldsymbol{\Sigma}\mathbf{W}\right)$。如果样本是 d 元正态的，则它投影到 k 元正态上，允许我们在很有希望的、低得多的维空间进行参数判别分析。因为 z_j 是不相关的，因此新的协方差矩阵将是对角的。如果它们被规范化，具有单位方差，则可以在这个新空间使用欧氏距离，导出简单的分类器。

实例 $\boldsymbol{x}^t$ 投影到 z-空间

$$\boldsymbol{z}^t = \mathbf{W}^T(\boldsymbol{x}^t - \boldsymbol{\mu})$$

当 $\mathbf{W}$ 是正交矩阵使得 $\mathbf{W}\mathbf{W}^T = \mathbf{I}$ 时，它可以逆投影到原来的空间

$$\hat{\boldsymbol{x}}^t = \mathbf{W}\boldsymbol{z}^t + \boldsymbol{\mu}$$

$\hat{\boldsymbol{x}}^t$ 是 $\boldsymbol{x}^t$ 从它在 z-空间中的表示的重构。我们知道，在所有正交线性投影中，PCA 最小化重构误差(reconstruction error)。重构误差是实例与它的从低维空间重构之间的距离：

$$\sum_t \| \hat{\boldsymbol{x}}^t - \boldsymbol{x} \|^2 \tag{6.12}$$

重构误差取决于考虑了多少个主成分。在视觉识别应用中，例如，人脸识别，显示 $\hat{\boldsymbol{x}}^t$ 使得我们能够可视化地检查 PCA 期间的信息损失。

PCA 是非监督的，并且不利用输出信息。它是一个一组(one-group)过程。然而，在分类情况下会有很多组，Karhunen-Loève 扩展(Karhunen-Loève expansion)允许利用类信息；例如，我们不是使用整个样本的协方差矩阵，而是估计类的协方差矩阵，取它们的平均(用先验加权)作为协方差矩阵，并使用它的本征向量。

在公共主成分(common principal component)中(Flury 1988)，我们假设对于每个类，主成分都是相同的，但是对于不同的类，这些成分的方差不同：

$$\mathbf{S}_i = \mathbf{C}\mathbf{D}_i\mathbf{C}^T$$

这种方法允许汇聚数据，并且是一种正则化方法，它的复杂度比所有类的公共协方差矩阵的方法小，同时仍允许 $\mathbf{S}_i$ 存在差别。一种相关的方法是柔性判别分析(flexible discriminant analysis)(Hastie、Tibshirani 和 Buja 1994)，它将数据线性投影到所有特征都是不相关的低维空间，再用最小距离分类。

114
~
115

6.4 因子分析

在 PCA 中，从原来的维 $x_i (i = 1, \cdots, d)$，我们形成一个新的变量集 $\boldsymbol{z}$，它是 x_i 的线性组合：

$$\boldsymbol{z} = \mathbf{W}^T(\boldsymbol{x} - \boldsymbol{\mu})$$

在因子分析(factor analysis，FA)中，我们假定有一个不可观测的潜在因子(latent factor) $z_j (j = 1, \cdots, k)$ 的集合，它在组合时生成 $\boldsymbol{x}$。因此，与 PCA 的方向相反(参见图 6-4)，其目标是通过较少数量的因子刻画观测变量之间的依赖性。

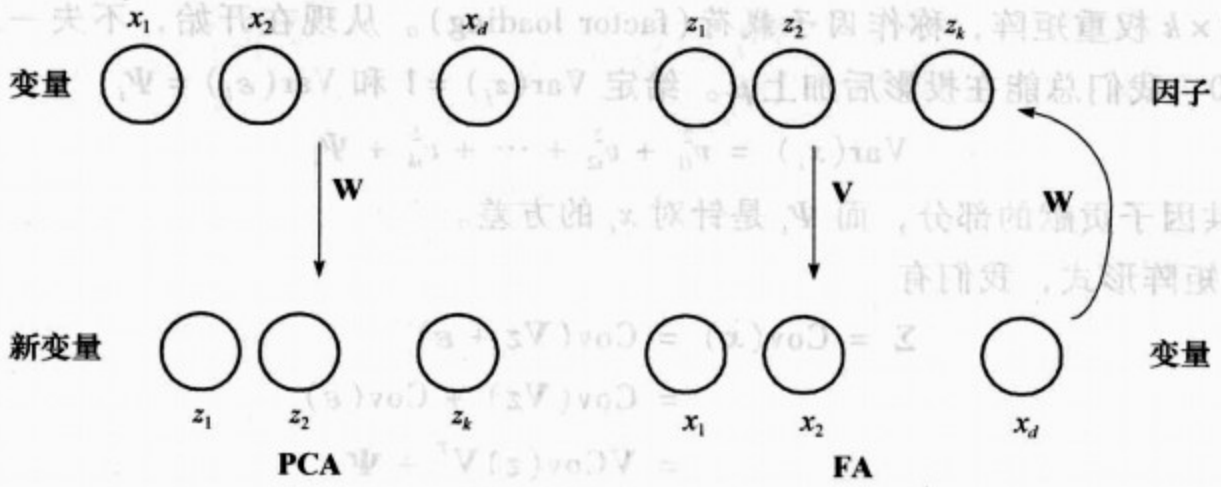


图 6-4 主成分分析方法产生新的变量，它们是原输入变量的线性组合。然而，在因子分析，我们假定存在一些因子，它们线性组合时产生输入变量

假设有一个变量组，它们之间具有高度相关性，而与其他所有变量具有很低的相关性。那么可能存在一个简单的潜在因子给出这些变量的起源。如果其他变量能够类似地分成子集，则少数因子就能够代表这些变量组。虽然因子分析总是把变量划分成因子簇，但是因子是否意味着什么，或是否真的存在，仍然是一个悬而未决的问题。

像 PCA 一样，FA 也是一个一组过程，并且是非监督的。目标是在一个更小的维空间中对数据建模而不丢失信息。在 FA 中，这用变量之间的相关性度量。

正如在 PCA 中一样，我们有样本 $\mathcal{X}=\{\boldsymbol{x}^t\}_t$，取自某个未知的概率密度，其中 $E[\boldsymbol{x}]=\boldsymbol{\mu}$，$\text{Cov}(\boldsymbol{x})=\boldsymbol{\Sigma}$。我们假定因子是单位正态的 $E[z_j]=0$，$\text{Var}(z_j)=1$，并且是不相关的 $\text{Cov}(z_i, z_j)=0$，$i\neq j$。为了说明什么是不能由因子解释的，每个输入存在一个附加的源，记作 ε_i。假定它具有 0 均值 $E[\varepsilon_i]=0$，和某个未知的方差 $\text{Var}(\varepsilon_i)=\Psi_i$。这些特殊的源之间是不相关的 $\text{Cov}(\varepsilon_i, \varepsilon_j)=0$，$i\neq j$，并且与因子也是不相关的 $\text{Cov}(\varepsilon_i, z_j)=0$，$\forall i, j$。

FA 假定每个输入维 $x_i(i=1, \cdots, d)$ 可以写成 $k<d$ 个因子 $z_j(j=1, \cdots, k)$ 的加权和，加上残差项(见图 6-5)：

$$x_i-\mu_i=v_{i1}z_1+v_{i2}z_2+\cdots+v_{ik}z_k+\varepsilon_i, \quad \forall i=1,\cdots,d$$

$$x_i-\mu_i=\sum_{j=1}^{k}v_{ij}z_j+\varepsilon_i \tag{6.13}$$

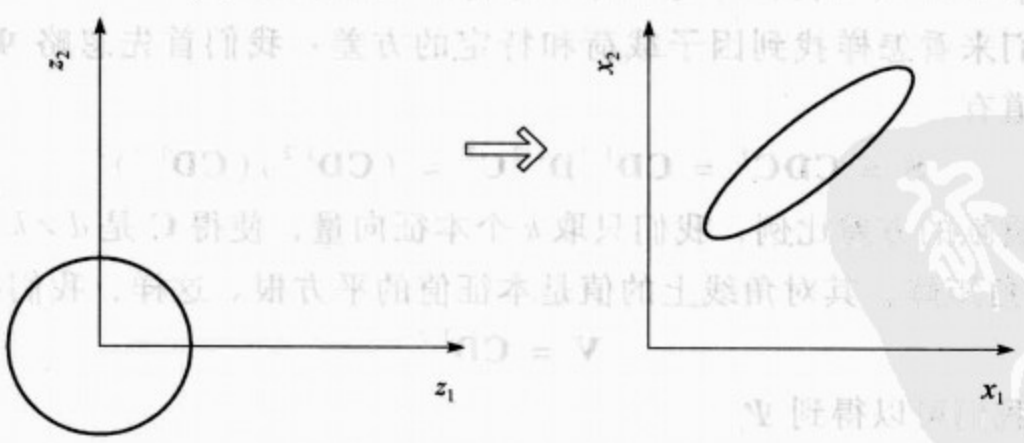


图 6-5 因子是独立的、单位正态的，它们被延伸、旋转和变换，以成为输入

这可以写成向量矩阵形式

$$\boldsymbol{x}-\boldsymbol{\mu}=\mathbf{V}\boldsymbol{z}+\boldsymbol{\varepsilon} \tag{6.14}$$

其中 $\mathbf{V}$ 是 $d \times k$ 权重矩阵，称作因子载荷(factor loading)。从现在开始，不失一般性，我们将假设 $\boldsymbol{\mu} = 0$；我们总能在投影后加上 $\boldsymbol{\mu}$。给定 $\text{Var}(z_j) = 1$ 和 $\text{Var}(\varepsilon_i) = \Psi_i$

$$\text{Var}(x_i) = v_{i1}^2 + v_{i2}^2 + \cdots + v_{ik}^2 + \Psi_i \tag{6.15}$$

$\Sigma_{j=1}^{k} v_{ij}^2$ 是公共因子贡献的部分，而 Ψ_i 是针对 x_i 的方差。

用向量矩阵形式，我们有

$$\begin{aligned} \boldsymbol{\Sigma} = \text{Cov}(\mathbf{x}) &= \text{Cov}(\mathbf{Vz} + \boldsymbol{\varepsilon}) \qquad (6.16) \\ &= \text{Cov}(\mathbf{Vz}) + \text{Cov}(\boldsymbol{\varepsilon}) \\ &= \mathbf{V}\text{Cov}(\mathbf{z})\mathbf{V}^T + \boldsymbol{\Psi} \\ &= \mathbf{V}\mathbf{V}^T + \boldsymbol{\Psi} \qquad (6.17) \end{aligned}$$

其中 $\boldsymbol{\Psi}$ 是对角矩阵，Ψ_i 在对角线上。由于因子是不相关的、单位正态的，因此我们有 $\text{Cov}(\mathbf{z}) = \mathbf{I}$。例如，对于两个因子

$$\text{Cov}(x_1, x_2) = v_{11}v_{21} + v_{12}v_{22}$$

如果 x_1 和 x_2 的协方差高，则它们通过一个因子相关。如果它是第一个因子，则 v_{11} 和 v_{21} 都高；如果它是第二个因子，则 v_{12} 和 v_{22} 都高。在这两种情况下，和 $v_{11}v_{21} + v_{12}v_{22}$ 都将会高。如果该协方差低，则 x_1 和 x_2 依赖于不同的因子，并且在和式的乘积中，一项高而另一项低，而它们的和低。

我们看到

$$\text{Cov}(x_1, z_2) = \text{Cov}(v_{12}z_2, z_2) = v_{12}\text{Var}(z_2) = v_{12}$$

因此，$\text{Cov}(\mathbf{x}, \mathbf{z}) = \mathbf{V}$，并且我们看到载荷用因子表示变量之间的相关性。

给定 $\boldsymbol{\Sigma}$ 的估计 $\mathbf{S}$，我们希望求解 $\mathbf{V}$ 和 $\boldsymbol{\Psi}$，满足

$$\mathbf{S} = \mathbf{V}\mathbf{V}^T + \boldsymbol{\Psi}$$

如果只有少量因子，即如果 $\mathbf{V}$ 只有少数几列，则因为 $\mathbf{V}$ 是 $d \times k$ 而 $\boldsymbol{\Psi}$ 有 d 个值，我们就能有一个关于 $\mathbf{S}$ 的简化结构，这样参数的数量从 d^2 减少到 $d \cdot k + d$。

因为 $\boldsymbol{\Psi}$ 是对角的，因此协方差由 $\mathbf{V}$ 表示。注意，PCA 不允许单独的 $\boldsymbol{\Psi}$ 并且试图考虑协方差和方差。当所有的 Ψ_i 相等时，即当 $\boldsymbol{\Psi} = \Psi\mathbf{I}$ 时，我们得到概率 PCA(probabilistic PCA)(Tipping 和 Bishop 1997)，而当 Ψ_i 为 0 时，得到传统的 PCA。

现在，让我们来看怎样找到因子载荷和特定的方差：我们首先忽略 $\boldsymbol{\Psi}$。然后，从它的谱分解，我们知道有

$$\mathbf{S} = \mathbf{CDC}^T = \mathbf{CD}^{1/2}\mathbf{D}^{1/2}\mathbf{C}^T = (\mathbf{CD}^{1/2})(\mathbf{CD}^{1/2})^T$$

116 ~ 118

其中，通过观察贡献的方差比例，我们只取 k 个本征向量，使得 $\mathbf{C}$ 是 $d \times k$ 的本征向量矩阵，而 $\mathbf{D}^{1/2}$ 是 $k \times k$ 对角矩阵，其对角线上的值是本征值的平方根。这样，我们有

$$\mathbf{V} = \mathbf{CD}^{1/2} \tag{6.18}$$

从(6.15)式我们可以得到 Ψ_i

$$\Psi_i = s_i^2 - \sum_{j=1}^{k} v_{ij}^2 \tag{6.19}$$

注意当 $\mathbf{V}$ 与任一正交矩阵(即具有 $\mathbf{TT}^T = \mathbf{I}$ 性质的矩阵)相乘，它就是另一个有效解，因此解不是唯一的。

$$\mathbf{S} = (\mathbf{VT})(\mathbf{VT})^T = \mathbf{VTT}^T\mathbf{V}^T = \mathbf{VIV}^T = \mathbf{VV}^T$$

如果 $\mathbf{T}$ 是正交矩阵，则到原点的距离不变。如果 $\boldsymbol{z} = \mathbf{T}\boldsymbol{x}$，则

$$\boldsymbol{z}^T\boldsymbol{z} = (\mathbf{T}\boldsymbol{x})^T(\mathbf{T}\boldsymbol{x}) = \boldsymbol{x}^T\mathbf{T}^T\mathbf{T}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{x}$$

乘以一个对角矩阵具有旋转坐标轴的效果，这允许我们选择最可解释的坐标集合(Rencher 1995)。在两维中，

$$\mathbf{T} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}$$

将坐标旋转 ϕ 度。有两种类型的旋转：在正交旋转中，旋转后因子仍然正交；在斜旋转中，允许因子变成相关的。旋转因子为每个变量在尽可能少的因子上给出最大载荷，使得因子可解释。然而，可解释性是主观的，不应该被用来强加个人对数据的偏见。

因子分析有两种用法：当我们找到载荷并且试图使用较少因子表示变量时，它可以用来提取知识。当 $k < d$ 时，它也可以被用来降低维度。我们已经看到了第一个是怎么做的。现在，让我们看一下因子分析怎样被用来降低维度。

当我们对维度归约感兴趣时，我们需要能够从 x_i 发现因子得分 z_j。我们希望找到载荷 w_{ji}，使得

$$z_j = \sum_{i=1}^{d} w_{ji}x_i + \varepsilon_i, \qquad j = 1, \cdots, k \tag{6.20}$$

119

其中 x_i 被中心化，具有均值 0。在向量形式，对于观测 t，这可以写作

$$\boldsymbol{z}^t = \mathbf{W}^T\boldsymbol{x}^t + \boldsymbol{\varepsilon}, \qquad \forall t = 1, \cdots, N$$

这是一个线性模型，有 d 个输入和 k 个输出。其转置可以写作

$$(\boldsymbol{z}^t)^T = (\boldsymbol{x}^t)^T\mathbf{W} + \boldsymbol{\varepsilon}^T, \qquad \forall t = 1, \cdots, N$$

给定我们有一个 N 个观测的样本，我们记

$$\mathbf{Z} = \mathbf{XW} + \Xi \tag{6.21}$$

其中 $\mathbf{Z}$ 是 $N \times k$ 个因子，$\mathbf{X}$ 是 $N \times d$ 个(中心化的)观测，而 Ξ 是 $N \times k$ 个 0 均值噪声。这是一个多输出的多元线性回归，并且我们从 5.8 节知道，可以求解 $\mathbf{W}$ 得到

$$\mathbf{W} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Z}$$

但是我们不知道 $\mathbf{Z}$；这是我们要计算的。我们在两边同时乘以和除以 $N-1$，得到

$$\begin{aligned} \mathbf{W} &= (N-1)(\mathbf{X}^T\mathbf{X})^{-1}\frac{\mathbf{X}^T\mathbf{Z}}{N-1} \\ &= \left(\frac{\mathbf{X}^T\mathbf{X}}{N-1}\right)^{-1}\frac{\mathbf{X}^T\mathbf{Z}}{N-1} \\ &= \mathbf{S}^{-1}\mathbf{V} \end{aligned} \tag{6.22}$$

并且把(6.22)式代入(6.21)式中，我们记作

$$\mathbf{Z} = \mathbf{XW} = \mathbf{XS}^{-1}\mathbf{V} \tag{6.23}$$

假定 $\mathbf{S}$ 是非奇异的。当 x_i 被规范化具有单位方差时，我们可以用 $\mathbf{R}$ 代替 $\mathbf{S}$。

对于维度归约，除了因子的可解释性，允许识别公共原因、简单解释、知识提取外，FA 与 PCA 相比并无优势。例如，在语音识别，$\boldsymbol{x}$ 对应声音信号，但是我们知道这是少数的

发音器官，即颚、舌、软腭、嘴唇和口腔(非线性)相互作用的结果，它们被适当的定位用来形成从肺部出来的气流，进而产生语音。如果语音信号可以转换到这个发音分析的空间，则语音识别就会非常容易。这是当前语音识别的研究方向之一。

120

6.5 多维定标

假设我们有 N 个点，并且给定每对点之间的距离 d_{ij}，$i, j=1, \cdots, N$。我们不知道这些点的确切坐标，也不知道它们的维度，以及距离是怎样计算的。多维定标(multidimensional scaling，MDS)是把这些点映射到低维(例如，两维)空间的方法，使得它们在两维空间中的欧氏距离尽可能接近在原空间中给定的距离 d_{ij}。这样，它需要一个从某个未知维度空间到低维空间，例如两维空间上的投影。

在典型的多维定标例子中，我们取城市之间的道路旅行距离，在应用 MDS 后，我们得到一张近似地图。这个地图被扭曲，在存在诸如高山和湖泊等地理障碍物的部分，道路旅行距离大大的偏离了直接的飞行距离(欧氏距离)，这个地图被拉伸，以便适应更长的距离(见图 6-6)。该地图以原点为中心，但是解仍然不是唯一的。我们可以得到任意的旋转和镜像版本。

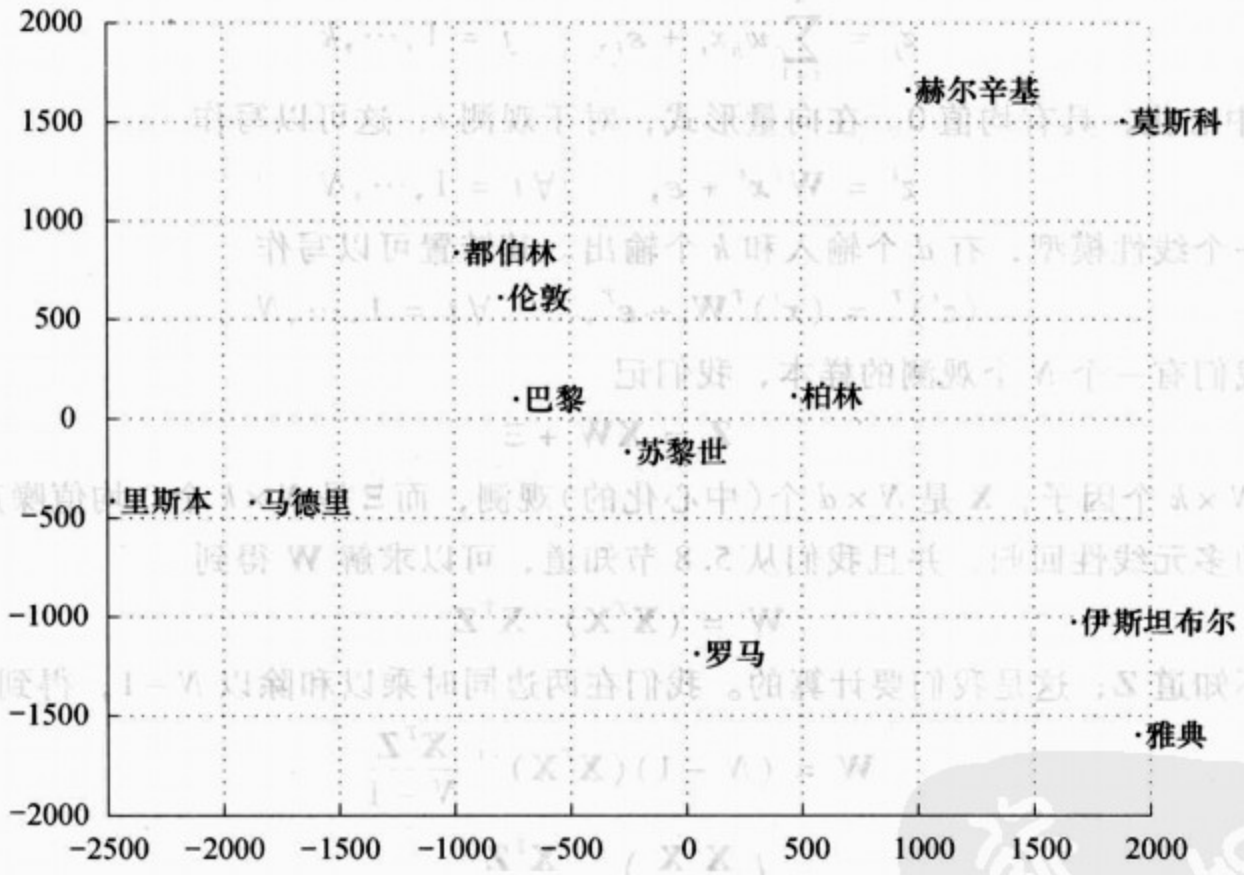


图 6-6 MDS 绘制的欧洲图。城市包括雅典、柏林、都柏林、赫尔辛基、伊斯坦布尔、里斯本、伦敦、马德里、莫斯科、巴黎、罗马和苏黎世。这些城市之间逐对道路旅行距离作为输入给出，并且 MDS 把它们放到两维的空间中，使得这些距离尽可能地被保持

可以使用 MDS 进行维度归约：通过计算 d 维 $\boldsymbol{x}$ 空间的逐对欧氏距离并把它作为 MDS 的输入。然后，MDS 把它投影到较低维的空间，以保持这些距离。

假设我们有通常的样本 $\mathcal{X}=\{\boldsymbol{x}^t\}_{t=1}^N$，其中 $\boldsymbol{x}^t \in \Re^d$。对于两个点 r 和 s，它们之间的平方

欧氏距离为

$$d_{rs}^2 = \| \boldsymbol{x}^r - \boldsymbol{x}^s \|^2 = \sum_{j=1}^{d} (x_j^r - x_j^s)^2 = \sum_{j=1}^{d} (x_j^r)^2 - 2\sum_{j=1}^{d} x_j^r x_j^s + \sum_{j=1}^{d} (x_j^s)^2$$
$$= b_{rr} + b_{ss} - 2b_{rs} \tag{6.24}$$

其中 b_{rs} 定义为

$$b_{rs} = \sum_{j=1}^{d} x_j^r x_j^s \tag{6.25}$$

为了约束这个解，我们把数据在原点中心化并假定

$$\sum_{t=1}^{N} x_j^t = 0, \quad \forall j = 1, \cdots, d$$

然后，在 r，s 和 r，s 二者上把(6.24)式加起来，并且定义

$$T = \sum_{t=1}^{N} b_{tt} = \sum_{t} \sum_{j} (x_j^t)^2$$

我们得到

$$\sum_{r} d_{rs}^2 = T + Nb_{ss}$$

$$\sum_{s} d_{rs}^2 = Nb_{rr} + T$$

$$\sum_{r} \sum_{s} d_{rs}^2 = 2NT$$

当我们定义

$$d_{\cdot s}^2 = \frac{1}{N}\sum_{r} d_{rs}^2, \quad d_{r\cdot}^2 = \frac{1}{N}\sum_{s} d_{rs}^2, \quad d_{\cdot\cdot}^2 = \frac{1}{N^2}\sum_{r}\sum_{s} d_{rs}^2$$

并使用(6.24)式时，我们得到

$$b_{rs} = \frac{1}{2}(d_{r\cdot}^2 + d_{\cdot s}^2 - d_{\cdot\cdot}^2 - d_{rs}^2) \tag{6.26}$$

现在，已经计算了 b_{rs} 并已知 $\boldsymbol{B} = \mathbf{X}\mathbf{X}^T$（如 6.25 式中的定义），我们寻找一个近似。从谱分解我们知道 $\mathbf{X} = \mathbf{C}\mathbf{D}^{1/2}$ 可以被用作 $\mathbf{X}$ 的一个近似，其中 $\mathbf{C}$ 是矩阵，其列是 $\mathbf{B}$ 的本征向量，而 $\mathbf{D}^{1/2}$ 是对角矩阵，其对角线是本征值的平方根。观察 $\mathbf{B}$ 的本征值，像我们在 PCA 和 FA 中所做的那样，我们确定比 d（和 N）低的维度 k。我们假设 $\boldsymbol{c}_j$ 是本征向量，其对应的本征值为 λ_j。注意 $\boldsymbol{c}_j$ 是 N 维的。于是，我们得到新的维

$$z_j^t = \sqrt{\lambda_j} c_j^t, \quad j = 1, \cdots, k, \quad t = 1, \cdots, N \tag{6.27}$$

也就是说，在标准化后，实例 t 的新坐标由本征向量 $\boldsymbol{c}_j$（$j=1$，…，k）的第 t 个元素给出。

已经证明(Chatfield 和 Collins 1980)$\mathbf{X}\mathbf{X}^T$($N \times N$)与 $\mathbf{X}^T\mathbf{X}$($d \times d$)的本征值相同，并且本征向量通过一个简单的线性变换相关。这说明 PCA 做了与 MDS 相同的工作，并且代价更低。在相关矩阵而不是在协方差矩阵上做 PCA 等价于用标准化的欧氏距离做 MDS，其中每个变量有单位方差。

在一般情况下，我们希望寻找一个映射 $\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x} \mid \theta)$，其中 $\boldsymbol{z} \in \Re^k$，$\boldsymbol{x} \in \Re^d$，并且 $\boldsymbol{g}(\boldsymbol{x} \mid \theta)$ 是根据参数 θ 的集合定义的从 d 维到 k 维的映射函数。前面我们讨论的经典的 MDS 对应线

性变换

$$z = g(x \mid \mathbf{W}) = \mathbf{W}^T x \tag{6.28}$$

但是在一般情况下，也可以使用非线性的映射；这称作 Sammon 映射(Sammon mapping)。在映射中的标准化误差称作 Sammon 应力(Sammon stress)，定义为

$$\begin{aligned} E(\theta \mid \mathcal{X}) &= \sum_{r,s} \frac{(\| z^r - z^s \| - \| x^r - x^s \|)^2}{\| x^r - x^s \|^2} \\ &= \sum_{r,s} \frac{(\| g(x^r \mid \theta) - g(x^s \mid \theta) \| - \| x^r - x^s \|^2)}{\| x^r - x^s \|^2} \end{aligned} \tag{6.29}$$

123

可以对 $g(\cdot \mid \theta)$使用任何回归方法，估计 θ 来最小化训练数据 $\mathcal{X}$ 上的应力。如果 $g(\cdot)$在 x 上是非线性的，这将对应一个非线性的维度归约。

在分类的情况下，我们可以在距离中包含类信息(见 Webb 1999)，如

$$d'_{rs} = (1 - \alpha) d_{rs} + \alpha c_{rs}$$

其中 c_{rs}是 x^r 和 x^s 所属类之间的“距离”。这个类间距离应该被主观地提供，而 α 用交叉确认优化。

6.6 线性判别分析

线性判别分析(linear discriminant analysis，LDA)是一种用于分类问题的维度归约的监督的方法。我们由两类来开始这个问题的讨论，然后推广到 $K>2$ 个类。

给定来自两个类 C_1 和 C_2 的样本，我们希望找到由向量 w 定义的方向，使得当数据投影到 w 上时，来自两个类的样本尽可能的分开。正如我们在前面看到的

$$z = w^T x \tag{6.30}$$

是 x 到 w 上的投影，因而也是一个从 d 维到 1 维的维度归约。

m_1 和 m_1 分别是 C_1 类样本在投影前和投影后的均值。注意 $m_1 \in \Re^d$ 而 $m_1 \in \Re$。我们有样本 $\mathcal{X} = \{x^t, r^t\}$，使得如果 $x^t \in C_1$ 则 $r^t = 1$，而如果 $x^t \in C_2$ 则 $r^t = 0$。

$$\begin{aligned} m_1 &= \frac{\sum_t w^T x^t r^t}{\sum_t r^t} = w^T m_1 \\ m_2 &= \frac{\sum_t w^T x^t (1 - r^t)}{\sum_t (1 - r^t)} = w^T m_2 \end{aligned} \tag{6.31}$$

来自 C_1 和 C_2 的样本投影后的散布(scatter)是

$$\begin{aligned} s_1^2 &= \sum_t (w^T x^t - m_1)^2 r^t \\ s_2^2 &= \sum_t (w^T x^t - m_2)^2 (1 - r^t) \end{aligned} \tag{6.32}$$

124

投影后，为了使两个类被很好的分开，我们希望均值尽可能远离，并且类实例散布在尽可能小的区域中。因此，我们希望 $|m_1 - m_2|$ 大，而 $s_1^2 + s_2^2$ 小(见图 6-7)。费希尔线性判别式(Fisher's linear discriminant)是最小化(6.33)式的 w。

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \tag{6.33}$$

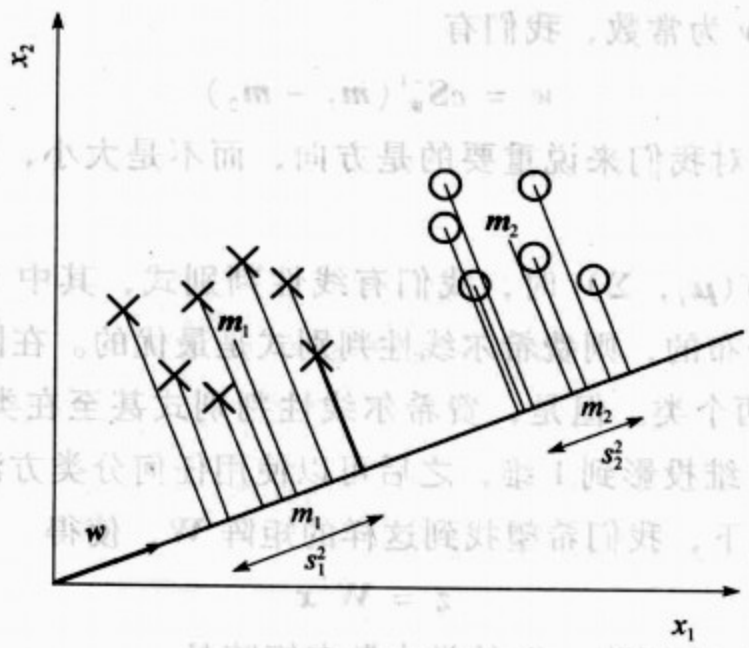


图 6-7 两维、两类的数据在 w 上的投影

重写分子，我们得到

$$\begin{aligned}(m_1 - m_2)^2 &= (w^T m_1 - w^T m_2)^2 \\ &= w^T (m_1 - m_2)(m_1 - m_2)^T w \\ &= w^T S_B w\end{aligned} \tag{6.34}$$

其中 $S_B = (m_1 - m_2)(m_1 - m_2)^T$ 是类间散布矩阵(between-class scatter matrix)。这个分子是投影后类实例在其均值周围散布的和，并且可以改写为

$$\begin{aligned}s_1^2 &= \sum_t (w^T x^t - m_1)^2 r^t \\ &= \sum_t w^T (x^t - m_1)(x^t - m_1)^T w r^t \\ &= w^T S_1 w\end{aligned} \tag{6.35}$$

125

其中

$$S_1 = \sum_t r^t (x^t - m_1)(x^t - m_1)^T \tag{6.36}$$

是 C_1 的类内散布矩阵(within-class scatter matrix)。$S_1 / \Sigma_t r^t$ 是 Σ_1 的估计。类似地，$s_2^2 = w^T S_2 w$，其中 $S_2 = \Sigma_t (1 - r^t)(x^t - m_2)(x^t - m_2)^T$，并且我们得到

$$s_1^2 + s_2^2 = w^T S_W w$$

其中，$S_W = S_1 + S_2$ 是类内散布的总和。注意，$s_1^2 + s_2^2$ 除以样本总数是汇聚数据的方差。(6.33)式可以改写为

$$J(w) = \frac{w^T S_B w}{w^T S_W w} = \frac{|w^T (m_1 - m_2)|^2}{w^T S_W w} \tag{6.37}$$

关于 w 取 J 的导数并令其等于0，我们得到

$$\frac{\boldsymbol{w}^T(\boldsymbol{m}_1-\boldsymbol{m}_2)}{\boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}}\left(2(\boldsymbol{m}_1-\boldsymbol{m}_2)-\frac{\boldsymbol{w}^T(\boldsymbol{m}_1-\boldsymbol{m}_2)}{\boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}}\mathbf{S}_W\boldsymbol{w}\right)=0$$

给定 $\boldsymbol{w}^T(\boldsymbol{m}_1-\boldsymbol{m}_2)/\boldsymbol{w}^T\mathbf{S}_W\boldsymbol{w}$ 为常数，我们有

$$\boldsymbol{w}=c\mathbf{S}_W^{-1}(\boldsymbol{m}_1-\boldsymbol{m}_2) \tag{6.38}$$

其中 c 是某个常数。因为对我们来说重要的是方向，而不是大小，所以我们可以取 $c=1$ 并找出 $\boldsymbol{w}$。

记住当 $p(\boldsymbol{x}\mid C_i)\sim\mathcal{N}(\boldsymbol{\mu}_i,\ \Sigma)$ 时，我们有线性判别式，其中 $\boldsymbol{w}=\Sigma^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)$，并且我们看到如果类是正态分布的，则费希尔线性判别式是最优的。在同样的假设下，我们还可以计算阈值 w_0，来分开两个类。但是，费希尔线性判别式甚至在类不是正态分布时也能使用。我们已经把样本从 d 维投影到 1 维，之后可以使用任何分类方法。

在 $K>2$ 个类的情况下，我们希望找到这样的矩阵 $\mathbf{W}$，使得

$$\boldsymbol{z}=\mathbf{W}^T\boldsymbol{x} \tag{6.39}$$

其中 $\boldsymbol{z}$ 是 k 维的，$\mathbf{W}$ 是 $d\times k$ 矩阵。C_i 的类内散布矩阵是

126

$$\mathbf{S}_i=\sum_t r_i^t(\boldsymbol{x}^t-\boldsymbol{m}_i)(\boldsymbol{x}^t-\boldsymbol{m}_i)^T \tag{6.40}$$

其中，如果 $\boldsymbol{x}^t\in C_i$ 则 $r_i^t=1$，否则为 0。总类内散布是

$$\mathbf{S}_W=\sum_{i=1}^K\mathbf{S}_i \tag{6.41}$$

当存在 $K>2$ 个类时，均值的散布根据它们在总均值周围的散布情况计算

$$\boldsymbol{m}=\frac{1}{K}\sum_{i=1}^K\boldsymbol{m}_i \tag{6.42}$$

而类间散布矩阵是

$$\mathbf{S}_B=\sum_{i=1}^K N_i(\boldsymbol{m}_i-\boldsymbol{m})(\boldsymbol{m}_i-\boldsymbol{m})^T \tag{6.43}$$

其中 $N_i=\Sigma_t r_i^t$。投影后的类间散布矩阵是 $\mathbf{W}^T\mathbf{S}_B\mathbf{W}$，而投影后的类内散布矩阵是 $\mathbf{W}^T\mathbf{S}_W\mathbf{W}$。它们都是 $k\times k$ 矩阵。我们希望第一个散布大；也就是说，在投影之后，在新的 k 维空间，我们希望类均值互相之间尽可能远离。我们希望第二个散布小；也就是说，在投影之后，我们希望来自同一个类的样本尽可能接近它们的均值。对于一个散布(或协方差)矩阵，散布的一个度量是行列式。记住该行列式是特征值的乘积，而特征值给出沿着它的本征向量(成分)的方差。因此，我们对最小化(6.44)式的矩阵 $\mathbf{W}$ 感兴趣

$$J(\mathbf{W})=\frac{|\mathbf{W}^T\mathbf{S}_B\mathbf{W}|}{|\mathbf{W}^T\mathbf{S}_W\mathbf{W}|} \tag{6.44}$$

$\mathbf{S}_W^{-1}\mathbf{S}_B$ 的最大的本征向量是解。$\mathbf{S}_B$ 是 K 个秩为 1 的矩阵 $(\boldsymbol{m}_i-\boldsymbol{m})(\boldsymbol{m}_i-\boldsymbol{m})^T$ 的和，并且它们之中只有 $K-1$ 个是独立的。因此，$\mathbf{S}_B$ 具有最大秩 $K-1$，并且我们取 $k=K-1$。这样，我们定义一个新的、较低的 $K-1$ 维空间，然后在那里构造判别式(见图 6-8)。虽然 LDA 使用类分离性作为它的好坏标准，但是在这个新空间里可以使用任意的分类方法来估计判别式。

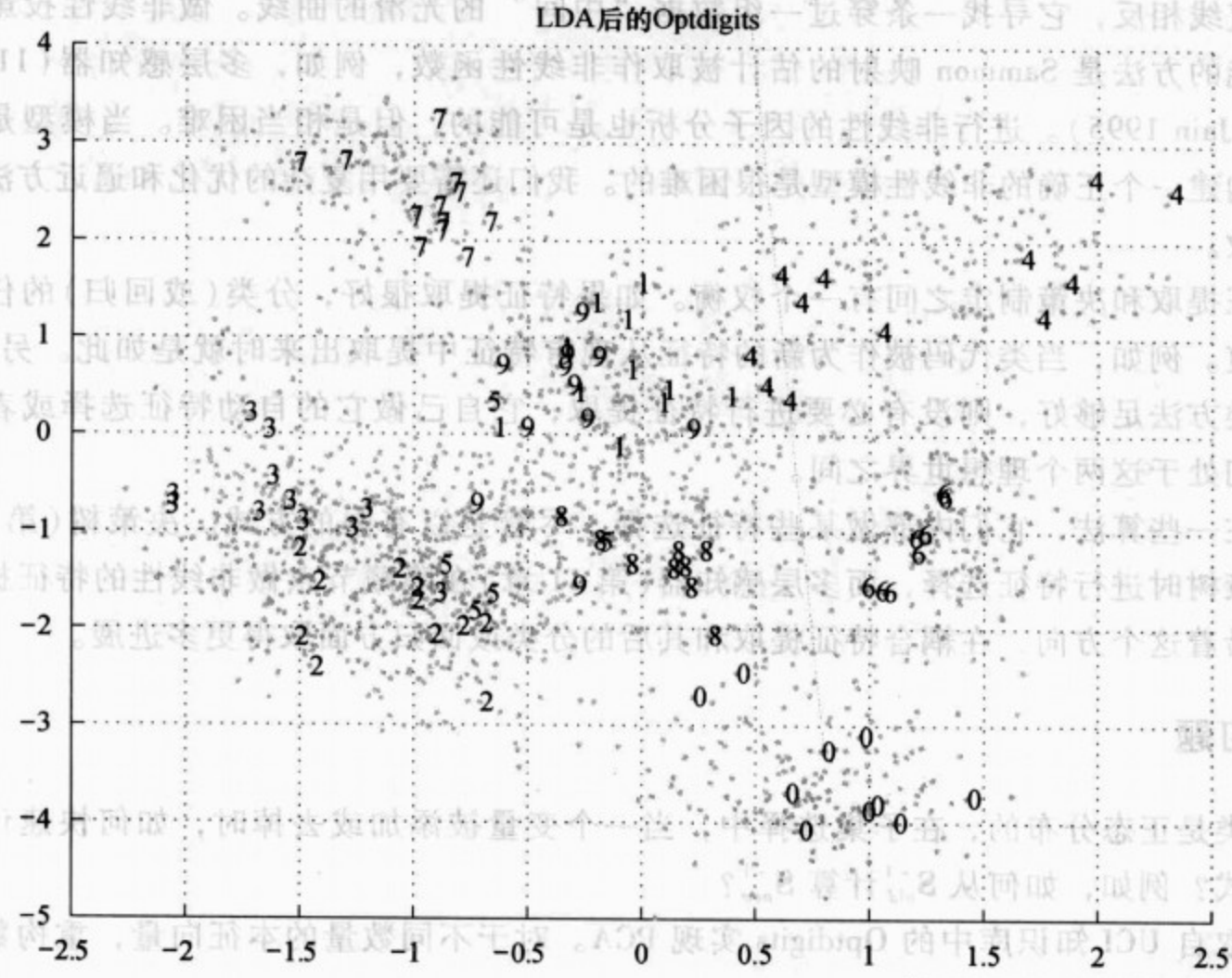


图 6-8 绘制在 LDA 找到的前两个维空间上的 Optdigits。与图 6-3 比较，正如期望的那样，我们看到 LDA 比 PCA 导致更好的类分离。即便在这个二维空间（有 9 个），我们也能看到不同类的分开的云团

6.7 注释

特征选择算法的一个综述在 Devijer 和 Kittler 1982 中给出。Miller 1990 讨论了回归中的子集选择。我们讨论的向前和向后搜索过程是局部搜索过程。Fukunaga 和 Narendra(1977)提出了一种分支和限制(branch 和 bound)过程。以更大的开销为代价，我们可以使用诸如模拟退火或者遗传算法这样的随机过程，在搜索空间中进行更广泛的搜索。对于分类，取代每步训练一个分类器并对它测试，我们可以使用像在线性判别式分析中使用的启发式方法，来度量新空间把类彼此分开的质量(McLachlan 1992)。

投影方法需要数值输入，并且离散变量应该用 0/1 哑变量表示，而子集选择可以直接使用离散输入。寻找特征向量和特征值是非常直接的，一个代码的例子在 Press 等 1992 中给出。因子分析是由英国的心理学家 Charles Spearman 引入的，用于发现智力的单个因素，来解释不同智力测试得分之间的联系。这种单个因子(称为 g)的存在性是讨论的课题。更多关于多维定标的信息可以在 Cox 和 Cox 1994 中找到。

127
ı
128

我们讨论的投影方法是批过程，因为它们要求在发现投影方向之前给定整个的样本。Mao 和 Jain(1995)讨论做 PCA 和 LDA 的在线过程，其中样例被逐个给出，并且更新随新实例的到达而进行。

本章讨论的线性投影方法具有局限性。在许多应用中，特征以非线性方式互相影响，需要非线性特征提取方法。主曲线(principal curve)(Hastie 和 Stuetzle 1989)允许非线性投影，

并且与直线相反，它寻找一条穿过一组数据“中间”的光滑的曲线。做非线性投影的另外一种可能的方法是 Sammon 映射的估计被取作非线性函数，例如，多层感知器(11.11 节)(Mao 和 Jain 1995)。进行非线性的因子分析也是可能的，但是相当困难。当模型是非线性的时，构建一个正确的非线性模型是很困难的。我们还需要用复杂的优化和逼近方法来求解模型参数。

特征提取和决策制定之间有一个权衡。如果特征提取很好，分类(或回归)的任务变得微不足道。例如，当类代码被作为新的特征从现有特征中提取出来时就是如此。另一方面，如果分类方法足够好，则没有必要进行特征提取；它自己做它的自动特征选择或者内部组合。我们处于这两个理想世界之间。

存在一些算法，它们内部做某些特征选择，尽管是以有限的方式。决策树(第 9 章)在产生决策树时进行特征选择，而多层感知器(第 11 章)在隐藏节点做非线性的特征提取。我
129 们期望沿着这个方向，在耦合特征提取和其后的分类或回归方面取得更多进展。

6.8 习题

1. 假定类是正态分布的，在子集选择中，当一个变量被添加或去掉时，如何快速计算新的判别式？例如，如何从 $\mathbf{S}_{old}^{-1}$ 计算 $\mathbf{S}_{new}^{-1}$？
2. 使用取自 UCI 知识库中的 Optdigits 实现 PCA。对于不同数量的本征向量，重构数字图像并计算重构误差(6.12)式。
3. 给定道路旅行距离作为输入，使用 MDS 绘制你所在州或国家的地图。
4. 在 Sammon 映射中，如果映射是线性的，即 $g(\boldsymbol{x}\mid\mathbf{W})=\mathbf{W}^{\mathrm{T}}\boldsymbol{x}$，如何计算最小化 Sammon 应力的 $\mathbf{W}$？

6.9 参考文献

Chatfield, C., and A. J. Collins. 1980. *Introduction to Multivariate Analysis*. London: Chapman and Hall.

Cox, T. F., and M. A. A. Cox. 1994. *Multidimensional Scaling*. London: Chapman and Hall.

Devijer, P. A., and J. Kittler. 1982. *Pattern Recognition: A Statistical Approach*. New York: Prentice-Hall.

Flury, B. 1988. *Common Principal Components and Related Multivariate Models*. New York: Wiley.

Fukunaga, K., and P. M. Narendra. 1977. "A Branch and Bound Algorithm for Feature Subset Selection." *IEEE Transactions on Computers* C-26: 917 – 922.

Hastie, T. J., and W. Stuetzle. 1989. "Principal Curves." *Journal of the American Statistical Association* 84: 502 – 516.

Hastie, T. J., R. J. Tibshirani, and A. Buja. 1994. "Flexible Discriminant Analysis by Optimal Scoring." *Journal of the American Statistical Association* 89: 1255 – 1270.

Mao, J., and A. K. Jain. 1995. "Artificial Neural Networks for Feature Extraction and Multivariate Data Projection." *IEEE Transactions on Neural Networks* 6: 296 – 317.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Miller, A. J. 1990. *Subset Selection in Regression*. London: Chapman and Hall.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1992. *Numerical Recipes in C*. Cambridge,

UK: Cambridge University Press.

Pudil, P., J. Novovičová, and J. Kittler. 1994. "Floating Search Methods in Feature Selection." *Pattern Recognition Letters* 15: 1119 – 1125.

Rencher, A. C. 1995. *Methods of Multivariate Analysis.* New York: Wiley.

Tipping, M. E., and C. M. Bishop. 1997. *Probabilistic Principal Components Analysis.* Technical Report NCRG/97/010, Neural Computing Research Group, Aston University, UK.

Turk, M., and A. Pentland. 1991. "Eigenfaces for Recognition." *Journal of Cognitive Neuroscience* 3: 71 – 86.

Webb, A. 1999. *Statistical Pattern Recognition.* London: Arnold.

130
≀
131

第7章 聚 类

在参数方法中，我们假设样本来自一个已知的分布。当这种假设站不住脚时，我们放宽该假设，并使用半参数方法，允许用混合分布估计输入样本。聚类方法允许从数据中学习混合分布。除了概率建模之外，我们还讨论向量量化和层次聚类。

7.1 引言

在第4章和第5章中，我们讨论了密度估计的参数方法，在那里我们假设样本$\boldsymbol{X}$取自某个参数族，例如高斯族。在参数分类中，这对应为类密度$p(\boldsymbol{x} \mid C_i)$假定某种密度。参数方法的优点是，给定一个模型，问题归结为少量参数的估计。对于密度估计，参数是密度的充分统计量。例如，对于高斯密度，参数为均值和协方差。

尽管参数方法使用频繁，但是对于假定并不成立的许多应用来说，假定一个严格的参数模型可能是偏倚根源。因此我们需要更灵活的模型。尤其是，假定高斯密度对应假定样本(例如一个类的实例)形成d-维空间中的单个分组，并且正如我们在第5章所看到的，该分组的中心和形状分别由均值协方差给定。

然而，在许多应用中，样本不是一个分组，而可能有多个分组。以手写字符识别为例：有两种风格书写数字7；美洲人的写法是‘7’，而欧洲人的写法是中间有一个水平杠(与欧洲人手写的、上面有一小划的‘1’以示区别)。在这种情况下，当样本包含来自两个洲的实例时，数字7应当表示成两个不相交的分组。如果每个分组用一个高斯分布表示，则该类可以用两个高斯分布的混合分布表示，每个高斯分布用于一种书写风格。

类似的例子是语音识别，其中由于不同的发音、口音、性别、年龄等，相同的词可能以不同的方法读出。这样，当没有单个、通用的原型时，为了统计上的正确性，应当在密度中表示所有这些不同的方法。

我们称这种方法为*半参数密度估计*(semiparametric density estimation)，因为我们仍然对样本中的每个分组假定一个参数模型。在第8章，我们将讨论非参数方法。当数据没有结构，甚至连混合模型都不能使用时，可以使用非参数方法。本章我们致力于密度估计，而将监督学习放到第12章。

7.2 混合密度

混合密度(mixture density)记作

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} p(\boldsymbol{x} \mid \mathcal{G}_i) P(\mathcal{G}_i) \tag{7.1}$$

其中$\mathcal{G}_i$是*混合分支*(mixture component)，也称*分组*(group)或*簇*(cluster)。$p(\boldsymbol{x} \mid \mathcal{G}_i)$是*支密度*

(component density)，而 $P(G_i)$ 是混合比例(mixture proportion)。分支数 k 是超级参数，应当预先指定。给定样本和 k，学习对应估计支密度和比例。当我们假定支密度遵守参数模型时，我们只需要估计它们的参数。如果支密度是多元高斯的，则我们有 $p(\boldsymbol{x}\mid G_i)\sim\mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$，而 $\Phi=\{P(G_i), \boldsymbol{\mu}_i, \Sigma_i,\}_{i=1}^{k}$ 是应当从独立同分布的样本 $\mathcal{X}=\{\boldsymbol{x}^t\}_t$ 中学习的参数。 134

参数分类是名副其实的混合模型，其中分组 G_i 对应类 C_i，支密度 $p(\boldsymbol{x}\mid G_i)$ 对应类密度 $p(\boldsymbol{x}\mid C_i)$，而 $P(G_i)$ 对应类先验 $P(C_i)$：

$$p(\boldsymbol{x})=\sum_{i=1}^{K}p(\boldsymbol{x}\mid C_i)P(C_i)$$

在这种监督情况下，我们知道有多少个分组，而学习参数却是平凡的，因为我们有类标号，即知道哪个实例属于哪个类(分支)。从第 5 章我们知道，给定样本 $\mathcal{X}=\{\boldsymbol{x}^t, \boldsymbol{r}^t\}_{t=1}^{N}$，其中如果 $\boldsymbol{x}^t\in C_i$ 则 $r_i^t=1$，否则 r_i^t 为 0，可以使用最大似然计算这些参数。当每个类都是高斯分布时，我们有混合高斯分布，并且参数估计为

$$\begin{aligned}\hat{P}(C_i)&=\frac{\Sigma_t r_i^t}{N}\\ \boldsymbol{m}_i&=\frac{\Sigma_t r_i^t\boldsymbol{x}^t}{\Sigma_t r_i^t}\\ \mathrm{S}_i&=\frac{\Sigma_t r_i^t(\boldsymbol{x}^t-\boldsymbol{m}_i)(\boldsymbol{x}^t-\boldsymbol{m}_i)^T}{\Sigma_t r_i^t}\end{aligned}\tag{7.2}$$

本章的不同是样本为 $\mathcal{X}=\{\boldsymbol{x}^t\}_t$，我们有非监督学习（unsupervised learning）问题。我们只有 $\boldsymbol{x}^t$ 而没有标号 $\boldsymbol{r}^t$，也就是说我们不知道 $\boldsymbol{x}^t$ 来自哪个分支。这样，我们应当估计二者：第一，我们应当估计给定实例所属的分支标号 r_i^t；第二，一旦我们估计了标号，我们就要估计给定实例集所属分支的参数。为此，我们首先讨论一种简单的聚类算法 k-均值，并在后面证明它是期望最大化（Expectation-Maximization）算法的一个特例。

7.3　k-均值聚类

假设我们有一幅图像，按 24 位/像素存放，而我们可能有多达 1 600 万种颜色。假定我们有 8 位/像素的彩色屏幕，只能显示 256 种颜色。我们想在 1 600 万种颜色中找出最佳的 256 种颜色，使得仅用了调色板中 256 种颜色的图像看上去尽可能接近原来的图像。这是颜色量化(color quantization)问题，其中我们从高分辨率映射到低分辨率。在一般情况下，目标是从连续空间映射到离散空间；这一过程称作向量量化(vector quantization)。 135

当然，我们总能均匀地进行量化，但是把映射表目指派到图像中不存在的颜色，或不给图像中频繁使用的颜色分配附加的表目会错失颜色映射。例如，如果图像是海景，则我们可望看到许多深浅不一的蓝色而可能不是红色。因此，颜色映射表目的分布应当尽可能接近地反映原来的密度，将更多的表目放在高密度区域，而丢弃没有数据的区域。

假定我们有样本 $\mathcal{X}=\{\boldsymbol{x}^t\}_{t=1}^{N}$。我们有 k 个参考向量(reference vector) $\boldsymbol{m}_j$，$j=1,\cdots,k$。在我们的颜色量化的例子中，$\boldsymbol{x}^t$ 是 24 位的图像像素值，$\boldsymbol{m}_j$ 是颜色映射表目，也是 24 位，

$k=256$。

暂时假定我们以某种方法得到了 $\boldsymbol{m}_j$ 的值；稍后我们将讨论如何学习它们。为了显示图像，给定像素 $\boldsymbol{x}^t$，我们用颜色映射中最相似的、满足下式的表目 $\boldsymbol{m}_i$ 表示它

$$\|\boldsymbol{x}^t-\boldsymbol{m}_i\|=\min_j\|\boldsymbol{x}^t-\boldsymbol{m}_j\|$$

也就是说，我们使用参考向量符号系统中最接近的值，而不是使用原始数据。$\boldsymbol{m}_i$ 又称码本向量(codebook vector)或码字(code word)，因为这是一个编码/解码过程(参见图7-1)：从 $\boldsymbol{x}^t$ 到 i 是使用编码本 $\boldsymbol{m}_i(i=1,\cdots,k)$对数据编码的过程，而在接收端，从 i 产生 $\boldsymbol{m}_i$ 是解码。量化也能压缩：例如，替代使用24位存储(或在通讯线上传输)每个 $\boldsymbol{x}^t$，我们可以只存储/传输它在颜色映射中的下标 i，使用8位索引1到256中的值，我们得到几乎为3的压缩率；存储/传输的也是颜色映射。

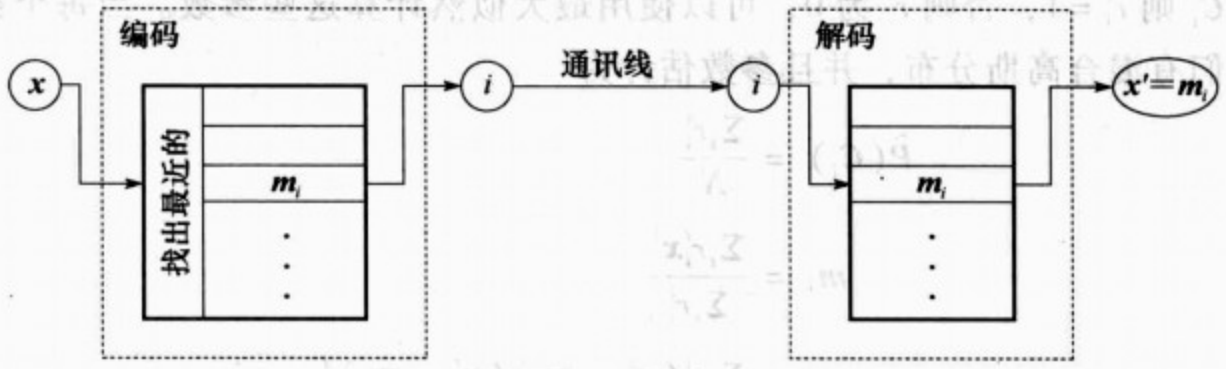


图7-1 给定 $\boldsymbol{x}$，编码器发送最近的码字的下标，而解码器使用接收到的下标产生码字 $\boldsymbol{x}'$。误差为 $\|\boldsymbol{x}'-\boldsymbol{x}\|^2$

让我们看看如何计算 $\boldsymbol{m}_i$：当 $\boldsymbol{x}^t$ 用 $\boldsymbol{m}_i$ 表示时，存在一个正比于距离 $\|\boldsymbol{x}^t-\boldsymbol{m}_i\|$ 的误差。为了使新图像看上去像原来的图像，我们应当针对所有的像素，使该距离尽可能小。总重构误差(reconstruction error)定义为

$$E(\{\boldsymbol{m}_i\}_{i=1}^k|X)=\sum_t\sum_i b_i^t\|\boldsymbol{x}^t-\boldsymbol{m}_i\|^2 \tag{7.3}$$

其中

$$b_i^t=\begin{cases}1 & \text{如果 } \|\boldsymbol{x}^t-\boldsymbol{m}_i\|=\min_j\|\boldsymbol{x}^t-\boldsymbol{m}_j\| \\ 0 & \text{否则}\end{cases} \tag{7.4}$$

最好的参考向量是最小化总重构误差的参考向量。b_i^t 也依赖 $\boldsymbol{m}_i$，并且我们不能解析地求解这个优化问题。对此，我们有一个称作 k-均值聚类(k-means clustering)的迭代过程：首先，我们以随机初始化的 $\boldsymbol{m}_i$ 开始。然后，在每次迭代中，我们先对每个 $\boldsymbol{x}^t$，使用(7.4)式计算估计标号(estimated labels) b_i^t；如果 b_i^t 为1，则 $\boldsymbol{x}^t$ 属于分组 $\boldsymbol{m}_i$。然后，一旦我们有了这些标号，我们就最小化(7.3)式。取它关于 $\boldsymbol{m}_i$ 的导数并令其等于0，我们得到

$$\boldsymbol{m}_i=\frac{\sum_t b_i^t\boldsymbol{x}^t}{\sum_t b_i^t} \tag{7.5}$$

参考向量被设置为它所代表的所有实例的均值。注意，除了用估计的标号 b_i^t 取代标号 r_i^t 外，这与(7.2)式中的均值公式相同。这是一个迭代过程，因为一旦我们计算了新的 $\boldsymbol{m}_i$，b_i^t

改变并且需要重新计算，这反过来又影响 $\boldsymbol{m}_i$。这个两步过程一直重复，直到 $\boldsymbol{m}_i$ 稳定(参见图 7-2)。k-均值算法的伪代码在图 7-3 中给出。

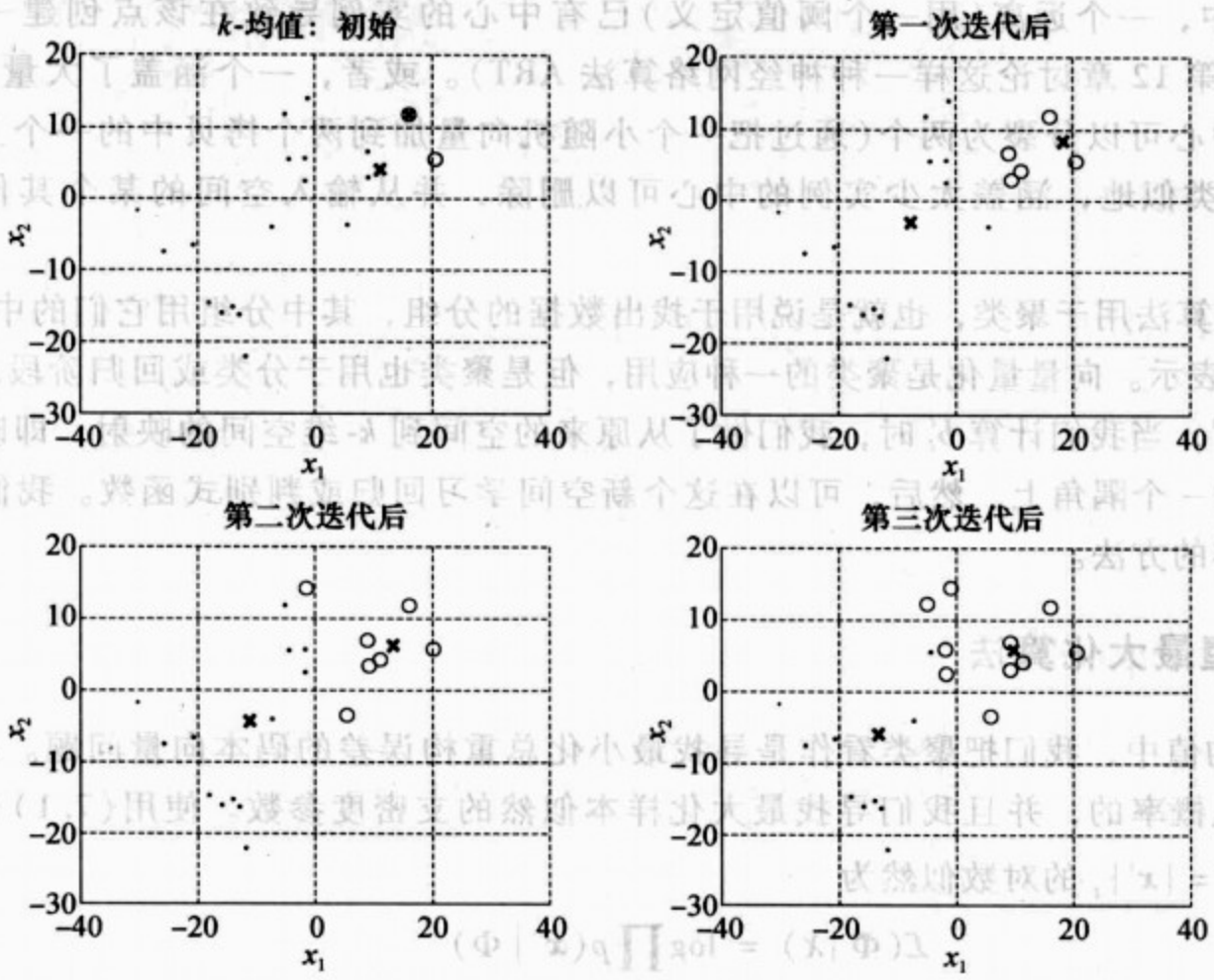


图 7-2 k-均值演变。叉指示中心位置。数据点根据最近的中心标记

初始化 $\boldsymbol{m}_i$，$i=1$，…，k；例如，将 $\boldsymbol{m}_i$ 初始化为 k 个随机的 $\boldsymbol{x}^t$

Repeat

For 所有的 $\boldsymbol{x}^t \in \mathcal{X}$

$$b_i^t \leftarrow \begin{cases} 1 & \text{如果 } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\| \\ 0 & \text{否则} \end{cases}$$

For 所有的 $\boldsymbol{m}_i$，$i=1$，…，k

$$\boldsymbol{m}_i \leftarrow \sum_t b_i^t \boldsymbol{x}^t / \sum_t b_i^t$$

Until $\boldsymbol{m}_i$ 收敛

图 7-3 k-均值算法

k-均值的一个缺点是它是一个局部搜索过程，并且最终的 $\boldsymbol{m}_i$ 高度依赖于初始的 $\boldsymbol{m}_i$。对于初始化，存在各种不同的方法：

- 可以简单地随机选择 k 个实例作为初始的 $\boldsymbol{m}_i$。
- 可以计算所有数据的均值，并将一些小随机向量加到均值上，得到 k 个初始的 $\boldsymbol{m}_i$。
- 可以计算主成分，将它的值域化分成 k 个相等的区间，将数据化分成 k 个分组，然后取这些分组的均值作为初始中心。

收敛后，所有的簇中心应当涵盖数据实例的某个子集并且是有用的，因此，最好将中心

初始化在有数据的地方。

还有一些算法动态地添加新中心或删除空的中心。在领导者聚类算法(leader cluster algorithm)中，一个远离(用一个阈值定义)已有中心的实例导致在该点创建一个新中心(我们将在第12章讨论这样一种神经网络算法ART)。或者，一个涵盖了大量实例($\Sigma_t b_i^t / N > \theta$)的中心可以分裂为两个(通过把一个小随机向量加到两个拷贝中的一个上，使得它们不同)。类似地，涵盖太少实例的中心可以删除，并从输入空间的某个其他部分重新开始。

k-均值算法用于聚类，也就是说用于找出数据的分组，其中分组用它们的中心(分组的典型代表)表示。向量量化是聚类的一种应用，但是聚类也用于分类或回归阶段之前的预处理。给定 $\boldsymbol{x}^t$，当我们计算 b_i^t 时，我们做了从原来的空间到 k-维空间的映射，即映射到 k-维超立方体的一个隅角上。然后，可以在这个新空间学习回归或判别式函数。我们将在第12章讨论这样的方法。

7.4 期望最大化算法

在 k-均值中，我们把聚类看作是寻找最小化总重构误差的码本向量问题。本节中，我们的方法是概率的，并且我们寻找最大化样本似然的支密度参数。使用(7.1)式的混合模型，样本 $\mathcal{X} = \{\boldsymbol{x}^t\}_t$ 的对数似然为

$$\mathcal{L}(\Phi | \mathcal{X}) = \log \prod_t p(\boldsymbol{x}^t | \Phi)$$

136
≀
139

$$= \sum_t \log \sum_{i=1}^k p(\boldsymbol{x}^t | \mathcal{G}_i) P(\mathcal{G}_i) \tag{7.6}$$

其中 Φ 包含先验概率 $P(\mathcal{G}_i)$ 和支密度 $p(\boldsymbol{x}^t | \mathcal{G}_i)$ 的有效统计量。不幸的是，我们不能解析地求解参数，而需要借助于迭代优化。

期望最大化(Expectation-Maximization，EM)算法(Dempster、Laird 和 Rubin 1977，Redner 和 Walker 1984)用于最大似然估计，其中问题涉及两组随机变量，其中一组 X 是可观测的，另一组 Z 是隐藏的。算法的目标是找到参数向量 Φ，它最大化 X 的观测值的似然 $\mathcal{L}(\Phi | \mathcal{X})$。但是，在不可行时，我们关联附加的隐藏变量(hidden variable) Z，并使用二者表示潜在的模型，最大化 X 和 Z 联合分布的似然，完全(complete)似然 $\mathcal{L}_c(\Phi | \mathcal{X}, \mathcal{Z})$。

由于 Z 值不是观测的，我们不能直接求解完全数据似然 $\mathcal{L}_c$。而是给定 $\mathcal{X}$ 和当前参数值 Φ^l(其中 l 是迭代次数)，我们求它的期望 $\mathcal{Q}$。这是算法的期望(E)步。然后，在最大化(M)步，我们寻找新的参数值 Φ^{l+1}，它最大化期望。这样

E 步：$\mathcal{Q}(\Phi | \Phi^l) = E[\mathcal{L}_c(\Phi | \mathcal{X}, \mathcal{Z}) | \mathcal{X}, \Phi^l]$

M 步：$\Phi^{l+1} = \arg\max_{\Phi} \mathcal{Q}(\Phi | \Phi^l)$

Dempster、Laird 和 Rubin(1977)证明增加 $\mathcal{Q}$ 意味着增加不完全似然

$$\mathcal{L}(\Phi^{l+1} | \mathcal{X}) \geq \mathcal{L}(\Phi^l | \mathcal{X})$$

在混合模型的情况下，隐藏变量是观测的源，即哪个观测属于哪个分支。如果这些被给定，例如作为监督学习的类标号被给定，我们就想知道调整哪些参数，以便拟合数据点。

EM 方法的执行过程如下：在 E 步，给定分支的当前知识，我们估计这些标号；而在 M 步，给定 E 步估计的标号，我们更新我们的类知识。这两步与 k-均值的两步相同：b_i^t 的计算(E 步)和 $\boldsymbol{m}_i$ 的重新估计(M 步)。

我们定义一个指示变量(indicator variable)向量 $\boldsymbol{z}^t = \{z_1^t, \cdots, z_k^t\}$，其中如果 $\boldsymbol{x}^t$ 属于簇 $\mathcal{G}_i$，则 $z_i^t = 1$，否则 z_i^t 为 0。z 是多项式分布，以先验概率 π_i 取自 k 个类，是 $P(\mathcal{G}_i)$的简写。 140
于是

$$P(\boldsymbol{z}^t) = \prod_{i=1}^{k} \pi_i^{z_i^t} \tag{7.7}$$

观测 $\boldsymbol{x}^t$ 的似然等于它的概率，被产生它的分支指定：

$$p(\boldsymbol{x}^t \mid \boldsymbol{z}^t) = \prod_{i=1}^{k} p_i(\boldsymbol{x}^t)^{z_i^t} \tag{7.8}$$

$p_i(\boldsymbol{x}^t)$是 $p(\boldsymbol{x}^t \mid \mathcal{G}_i)$的简写。联合密度为

$$p(\boldsymbol{x}^t, \boldsymbol{z}^t) = P(\boldsymbol{z}^t) p(\boldsymbol{x}^t \mid \boldsymbol{z}^t)$$

而独立同分布的样本 $\mathcal{X}$ 的完全数据似然为

$$\begin{aligned}
\mathcal{L}_c(\Phi \mid \mathcal{X}, \mathcal{Z}) &= \log \prod_t p(\boldsymbol{x}^t, \boldsymbol{z}^t \mid \Phi) \\
&= \sum_t \log p(\boldsymbol{x}^t, \boldsymbol{z}^t \mid \Phi) \\
&= \sum_t \log P(\boldsymbol{z}^t \mid \Phi) + \log p(\boldsymbol{x}^t \mid \boldsymbol{z}^t, \Phi) \\
&= \sum_t \sum_i z_i^t [\log \pi_i + \log p_i(\boldsymbol{x}^t \mid \Phi)]
\end{aligned}$$

E 步：我们定义

$$\begin{aligned}
\mathcal{Q}(\Phi \mid \Phi^l) &\equiv E[\log P(X, Z) \mid \mathcal{X}, \Phi^l] \\
&= E[\mathcal{L}_c(\Phi \mid \mathcal{X}, \mathcal{Z}) \mid \mathcal{X}, \Phi^l] \\
&= \sum_t \sum_i E[Z_i^t \mid \mathcal{X}, \Phi^l][\log \pi_i + \log p_i(\boldsymbol{x}^t \mid \Phi^l)]
\end{aligned}$$

其中

$$\begin{aligned}
E[z_i^t \mid \mathcal{X}, \Phi^l] &= E[z_i^t \mid \boldsymbol{x}^t, \Phi^l] && \boldsymbol{x}^t \text{ 是独立同分布} \\
&= P(z_i^t = 1 \mid \boldsymbol{x}^t, \Phi^l) && z_i^t \text{ 是 0/1 随机变量} \\
&= \frac{p(\boldsymbol{x}^t \mid z_i^t = 1, \Phi^l) P(z_i^t = 1 \mid \Phi^l)}{p(\boldsymbol{x}^t \mid \Phi^l)} && \text{贝叶斯规则} \\
&= \frac{p_i(\boldsymbol{x}^t \mid \Phi^l) \pi_i}{\Sigma_j p_j(\boldsymbol{x}^t \mid \Phi^l) \pi_j} \qquad 141 \\
&= \frac{p(\boldsymbol{x}^t \mid \mathcal{G}_i, \Phi^l) P(\mathcal{G}_i)}{\Sigma_j p(\boldsymbol{x}^t \mid \mathcal{G}_j, \Phi^l) P(\mathcal{G}_j)} \\
&= P(\mathcal{G}_i \mid \boldsymbol{x}^t, \Phi^l) \equiv h_i^t
\end{aligned} \tag{7.9}$$

我们看到隐藏变量的期望值 $E[z_i^t]$是 $\boldsymbol{x}^t$ 被分支$\mathcal{G}_i$ 产生的后验概率。因为这是概率，所以它在 0 和 1 之间，并且与 k-均值的 0/1“硬”标号不同，它是“软”标号。

M 步：我们最大化 $\mathcal{Q}$，得到下一组参数值 Φ^{l+1}：

$$\Phi^{l+1} = \arg\max_{\Phi} \mathcal{Q}(\Phi \mid \Phi^l)$$

其中

$$\begin{aligned}\mathcal{Q}(\Phi \mid \Phi^l) &= \sum_t \sum_i h_i^t [\log \pi_i + \log p_i(\boldsymbol{x}^t \mid \Phi^l)] \\ &= \sum_t \sum_i h_i^t \log \pi_i + \sum_t \sum_i h_i^t \log p_i(\boldsymbol{x}^t \mid \Phi^l)\end{aligned} \tag{7.10}$$

第二项独立于 π_i，并且作为拉格朗日，使用约束 $\Sigma_i \pi_i = 1$，我们求解

$$\nabla_{\pi_i} \sum_t \sum_i h_i^t \log \pi_i - \lambda \left(\sum_i \pi_i - 1 \right) = 0$$

并且得到

$$\pi_i = \frac{\Sigma_t h_i^t}{N} \tag{7.11}$$

这类似于(7.2)式的先验计算。

类似地，(7.10)式的第一项独立于分支，并且可以在估计分支的参数时丢弃。我们解

$$\nabla_{\Phi} \sum_t \sum_i h_i^t \log p_i(\boldsymbol{x}^t \mid \Phi) = 0 \tag{7.12}$$

如果我们假定高斯分支 $\hat{p}_i(\boldsymbol{x}^t \mid \Phi) \sim \mathcal{N}(\boldsymbol{m}_i, \mathbf{S}_i)$，则 M 步为

$$\boldsymbol{m}_i^{l+1} = \frac{\Sigma_t h_i^t \boldsymbol{x}^t}{\Sigma_t h_i^t} \tag{7.13}$$

142

$$\mathbf{S}_i^{l+1} = \frac{\Sigma_t h_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i^{l+1})(\boldsymbol{x}^t - \boldsymbol{m}_i^{l+1})^T}{\Sigma_t h_i^t}$$

这里，对于 E 步的高斯分支，我们计算

$$h_i^t = \frac{\pi_i \mid \mathbf{S}_i \mid^{-1/2} \exp[-(1/2)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T \mathbf{S}_i^{-1} (\boldsymbol{x}^t - \boldsymbol{m}_i)]}{\Sigma_j \pi_j \mid \mathbf{S}_j \mid^{-1/2} \exp[-(1/2)(\boldsymbol{x}^t - \boldsymbol{m}_j)^T \mathbf{S}_j^{-1} (\boldsymbol{x}^t - \boldsymbol{m}_j)]} \tag{7.14}$$

又一次，(7.13)与(7.2)式之间的相似性并非偶然；估计的软标号 h_i^t 取代了实际(未知的)标号 r_i^t。

EM 用 k-均值初始化。在几次 k-均值迭代后，我们得到中心 $\boldsymbol{m}_i$ 的估计，并且使用被每个中心涵盖的实例，我们估计 $\mathbf{S}_i$ 和 $\Sigma_t b_i^t / N$ 得到 π_i。从那之后，我们运行 EM，如图 7-4 所示。

正如参数分类(5.5 节)，使用小样本和高维度，我们可以通过化简假设来正则化。当 $\hat{p}_i(\boldsymbol{x}^t \mid \Phi) \sim \mathcal{N}(\boldsymbol{m}_i, \mathbf{S})$ 时，在共享协方差矩阵的情况下，(7.12)式化简为

$$\min_{\boldsymbol{m}_i, \mathbf{S}} \sum_t \sum_i h_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)^T \mathbf{S}^{-1} (\boldsymbol{x}^t - \boldsymbol{m}_i) \tag{7.15}$$

当 $\hat{p}_i(\boldsymbol{x}^t \mid \Phi) \sim \mathcal{N}(\boldsymbol{m}_i, s^2\mathbf{I})$ 时，在共享对角矩阵的情况下，我们有

$$\min_{\boldsymbol{m}_i, s} \sum_t \sum_i h_i^t \frac{\| \boldsymbol{x}^t - \boldsymbol{m}_i \|^2}{s^2} \tag{7.16}$$

这是我们在 k-均值聚类中定义的重构误差(7.3)式。现在，不同的是

$$h_i^t = \frac{\exp[-(1/2s^2) \| \boldsymbol{x}^t - \boldsymbol{m}_i \|^2]}{\Sigma_j \exp[-(1/2s^2) \| \boldsymbol{x}^t - \boldsymbol{m}_j \|^2]} \tag{7.17}$$

是 0 和 1 之间的概率。k-均值聚类中的 b_i^t 做 0/1 硬决策，而 h_i^t 是软标号，它以一定概率将输入指派到簇中。当使用 h_i^t 而不是 b_i^t 时，实例对所有分支的参数更新都有贡献，对每个分支以一定的概率。当实例靠近两个中心的中点时，这特别有用。这样，我们看到 k-均值聚类是 EM 用于高斯混合模型的特例，假定输入是独立的、具有相等和共享的方差，并且标号是“硬的”。k-均值用圆覆盖输入密度，而 EM 一般用任意形状和任意方向的椭圆。

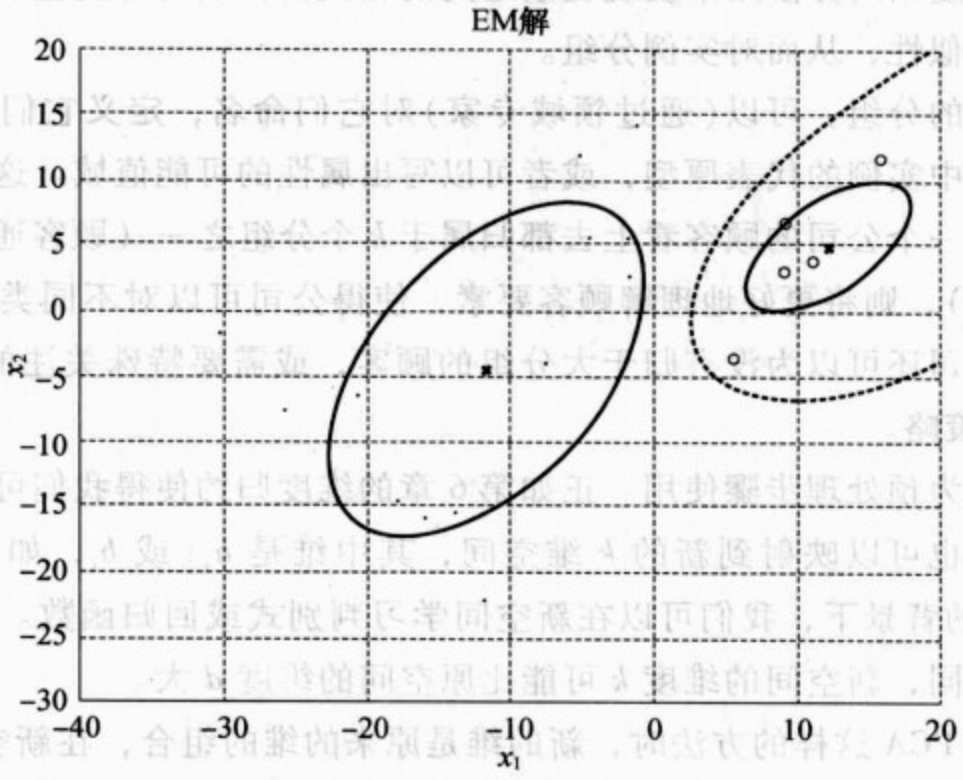


图 7-4　数据点和被 EM 拟合的高斯分布，被图 7-2 的一个 k-均值迭代初始化。不像 k-均值，EM 允许估计协方差矩阵。图中显示了被较大的 h_i 标记的数据点、估计的高斯密度的围线和 $h_i = 0.5$ 的分离曲线（虚线）

7.5　潜在变量混合模型

当全协方差矩阵与高斯混合分布一起使用时，即使没有奇异性，如果输入维度很高而样本很小，仍然有过分拟合的危险。为了减少参数的个数而假定具有共同的协方差矩阵可能并不正确，因为簇实际上可能具有不同的形状。假定对角矩阵可能更危险，因为这排出了所有的相关性。另一种选择是在簇中做维归约。这减少了参数个数，但仍然捕获相关性。自由参数的数量通过约化空间的维度控制。

当我们在簇中做因子分析时，我们寻找产生簇中数据的潜在变量（latent variable）或隐藏变量（hidden variable）或因子（factor）（Bishop 1999）：

$$p(\boldsymbol{x}^t \mid \mathcal{G}_i) \sim \mathcal{N}(\boldsymbol{m}_i, \mathbf{V}_i\mathbf{V}_i^T + \boldsymbol{\psi}_i) \tag{7.18}$$

143 ~ 144

其中 $\mathbf{V}_i$ 和 $\boldsymbol{\psi}_i$ 是簇 $\mathcal{G}_i$ 的因子载荷和特定方差。Rubin 和 Thayer（1982）给出了因子分析的 EM 方法。可以把它推广到混合模型，找到混合因子分析方法（mixtures of factor analyzers）（Ghahramani 和 Hinton 1997）。类似地，也可以在分组中做 PCA，称作混合概率主成分分析方法（mixtures of probabilistic principal component analyzer）（Tipping 和 Bishop 1999）。

当然，我们可以使用 EM 学习 $\mathbf{S}_i$，然后分别在每个簇上做 FA 或 PCA，但是做 EM 更好，

因为它将两个步骤结合在一起，并做软划分。每个实例对所有分组的潜在变量的计算都有贡献，权重为 h_i^t。

7.6 聚类后的监督学习

像第6章讨论的维度归约方法一样，聚类可以用于两个目的：它可以用来探查数据，理解数据的结构。维度归约方法用来发现变量之间的相关性，从而对变量分组。聚类方法用来发现实例之间的相似性，从而对实例分组。

如果找到这样的分组，可以(通过领域专家)对它们命名，定义它们的属性。我们可以选组均值作为分组中实例的代表原型，或者可以写出属性的可能值域。这可以更简单地描述数据。例如，如果一个公司的顾客看上去都归属于 k 个分组之一（顾客通过人口统计属性和与公司的交易勾画），则将更好地理解顾客要素，使得公司可以对不同类型的顾客使用不同的策略。同样，公司还可以为没有归于大分组的顾客，或需要特殊关注的顾客(例如，定期买卖的顾客)制定策略。

聚类也常常作为预处理步骤使用。正如第6章的维度归约使得我们可以映射到新空间一样，聚类后，我们也可以映射到新的 k 维空间，其中维是 h_i(或 b_i，如果损失一些信息的话)。在监督学习的背景下，我们可以在新空间学习判别式或回归函数。然而，与诸如 PCA 等维度归约方法不同，新空间的维度 k 可能比原空间的维度 d 大。

145

当我们使用像 PCA 这样的方法时，新的维是原来的维的组合，在新空间表示任意实例，所有的维都有贡献，即所有的 z_j 非零。在使用像聚类这种方法的情况下，新的维是局部定义的，存在很多新维 b_j，但是它们之中只有一个(或几个，如果使用 h_j)具有非零值。在前一种情况下，存在少量维，但都对表示起作用；在后一种情况下，存在许多维，但只有少量起作用。

在监督学习前面使用非监督聚类或维度归约的优点是，后者不需要标记的数据。标记数据的开销很大。我们可以使用大量未标记的数据学习簇参数，然后使用少量标记的数据在第二阶段学习分类或回归。非监督学习又称“学习通常发生的事”(Barrow 1989)。当后随监督学习时，我们先学习通常发生的事，然后学习它意味什么。我们将在第12章讨论这种方法。

对于分类，当每个类都是一个由大量分支组成的混合模型时，整个密度是混合的混合密度(mixture of mixtures)：

$$p(\boldsymbol{x} \mid C_i) = \sum_{j=1}^{k_i} p(\boldsymbol{x} \mid \mathcal{G}_{ij}) P(\mathcal{G}_{ij})$$

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} p(\boldsymbol{x} \mid C_i) P(C_i)$$

其中 k_i 是组成 $p(\boldsymbol{x} \mid C_i)$ 的分支数，$\mathcal{G}_{ij}$ 是类 i 的分支 j。正如我们前面所讨论的，分别为每个类学习分支的参数(或许在正则化之后)。这比用许多分支拟合所有的类的数据，然后用类标记它们的方法好。

7.7 层次聚类

我们从统计学观点讨论了聚类，将聚类看作用一个混合模型拟合数据，或找出最小化重构误差的码字。还有一些聚类方法，它们只使用实例之间的相似性，而对数据没有其他要求；目标是找出分组，使得在同一个分组中的对象比在不同分组中的对象更相似。这种方法通过层次聚类(hierarchical clustering)实现。 146

这需要使用定义在实例间的相似性度量，或等价地，定义距离度量。通常使用欧氏距离，其中我们需要确保所有的属性都具有相同的尺度。欧氏距离是闵可夫斯基距离(Minkowksi distance)的特例，其中 $p=2$：

$$d_m(\boldsymbol{x}^r,\boldsymbol{x}^s) = \left[\sum_{j=1}^{d}(x_j^r - x_j^s)^p\right]^{1/p}$$

城市块距离(city-block distance)容易计算

$$d_{cb}(\boldsymbol{x}^r,\boldsymbol{x}^s) = \sum_{j=1}^{d}|x_j^r - x_j^s|$$

凝聚聚类(agglomerative clustering)算法从 N 个分组开始，每个分组最初只包含一个训练实例，重复合并相似的分组形成较大的分组，直到只有一个分组。分裂聚类(divisive clustering)以相反的方向，从单个分组开始，并将较大的分组分裂成较小的分组，直到每个分组包含单个实例。

在凝聚算法的每次迭代，我们选择两个最近的分组合并。在单链接聚类(single-link clustering)中，距离定义为两个分组的所有可能元素对之间的最小距离：

$$d(\mathcal{G}_i,\mathcal{G}_j) = \min_{\boldsymbol{x}^r\in\mathcal{G}_i,\boldsymbol{x}^s\in\mathcal{G}_j} d(\boldsymbol{x}^r,\boldsymbol{x}^s) \tag{7.19}$$

考虑一个加权的完全连接的图，顶点对应实例，顶点之间的边的权重对应实例之间的距离。单链接方法对应构造该图的最小生成树。

在全链接聚类(complete-link clustering)中，两个分组之间的距离取所有可能对之间的最大距离：

$$d(\mathcal{G}_i,\mathcal{G}_j) = \max_{\boldsymbol{x}^r\in\mathcal{G}_i,\boldsymbol{x}^s\in\mathcal{G}_j} d(\boldsymbol{x}^r,\boldsymbol{x}^s) \tag{7.20}$$

这两种是最频繁使用的、用于选择最近的分组合并的度量。其他可能的选择是使用所有可能点对之间平均距离的平均链接方法，度量两个分组质心(均值)之间距离的质心距离。 147

一旦运行了凝聚方法，结果通常被绘制成一个称作系统树图(dendrogram)的层次结构。这是一棵树，其中树叶对应实例，按照它们合并的次序分组。一个例子在图 7-5 中给出。树可以在任意水平截断，得到期望个数的分组。

单链接和全链接方法用不同的方法计算分组之间的距离，这影响聚类结果和系统树图：在单链接方法中，两个实例在水平 h 合并为一组，如果它们之间的距离小于 h，或者存在一个中间实例序列使得相继实例之间的距离小于 h。另一方面，在全链接中，一个分组中的所有实例之间的距离都小于 h。单链接簇可能因这种“链接”效应而拉长。(在图 7-5 中，如果在 e 和 c 中间有一个实例会怎么样?)全链接簇趋向于更紧凑。

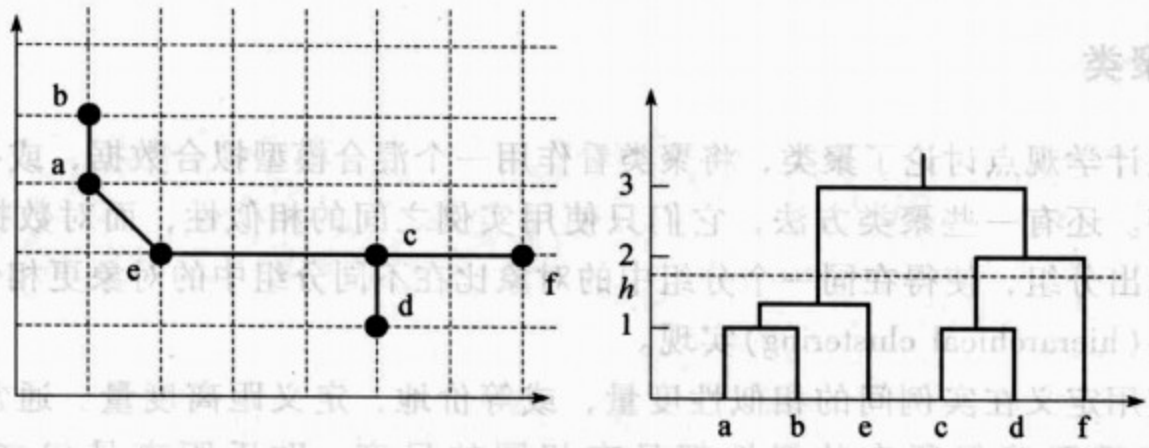


图 7-5 二维数据集和展示单链接聚类结果的系统树图。注意，树的树叶被排序使得分支不交叉。树在期望的值 h 上截断以得到簇

7.8 选择簇个数

像其他学习方法一样，聚类也有自己的调整复杂度的控制参数；这就是簇数 k。给定 k，聚类总是找出 k 个中心，不管它们是实际上有意义的分组，还是我们使用的算法强加的分组。存在多种调整 k 的方法：

- 在某些诸如颜色量化的应用中，k 由应用确定。
- 使用 PCA 在二维平面绘制数据可能用来发现数据的结构和数据中的簇数。
- 增量方法可能有助于确定 k：设置允许的最大距离等价于设置每个实例的允许的最大重构误差。
- 在某些实际应用中，分组确认可以人工地进行，即检查簇是否实际上对数据中有意义的分组编码。例如，在数据挖掘应用中，领域专家可以做这项工作。在颜色量化中，我们可以目视检查图像，检查它的质量(尽管我们的眼睛和大脑并不逐个像素分析图像)。

依赖于我们使用的聚类方法类型，我们可以将重构误差或对数似然作为 k 的函数绘制图形，并找出“拐点”。足够大的 k 之后，算法将开始分裂分组，在这种情况下，重构误差将不会大幅度降低，对数似然将不会大幅度提高。类似地，在层次聚类，通过观察水平之间的差，我们可以决定好的划分。

7.9 注释

混合模型在统计学中被频繁使用。专门的教科书包括 Titterington、Smith 和 Makov (1985)，McLachlan 和 Basford(1988)的书。McLachlan and Krishnan(1997)讨论了 EM 算法的最近进展、如何加快它的收敛性和各种变形。在信号处理过程中，k-均值称作 *Linde-Buzo-Gray*(LBG)算法(Gersho 和 Gray 1992)。k-均值频繁地用于统计学和信号处理的各种应用中，并且具有许多变形，其中之一是模糊 k-均值(fuzzy k-means)。输入与分支的模糊关系也是一个 0 和 1 之间的数(Bezdek 和 Pal 1995)。Alpaydın(1998)比较了 k-均值、模糊 k-均值和高斯混合模型上的 EM。Xu 和 Jordan(1996)给出了 EM 与学习高斯混合模型的其他学习算法的比较。在小样本上，另一种简化假设的方法是使用贝叶斯方法(Ormoneit 和 Tresp 1996)。Moer-

148 ~ 149

land(1999)在一组分类问题上比较了高斯混合模型和潜在变量混合模型，用实验展示了潜在变量模型的优点。Jain 和 Dubes(1988)是一本关于聚类的书，而 Jain、Murty 和 Flynn(1999)是一篇关于聚类的综述。

7.10 习题

1. 在图像压缩中，k-均值可以这样使用：图像被划分成 $c \times c$ 个窗口，并且这些 c^2 维向量构成样本。对于给定的 k(通常是 2 的幂)，我们做 k-均值聚类。参考向量和每个窗口的下标通过通信线路发送。在接收端，通过使用下标读取参考向量表来重构图像。写一个计算机程序，对于不同的 k 和 c 值做这件事。对每种情况，计算重构误差和压缩率。
2. 我们可以做 k-均值聚类，划分实例，然后分别在每个分组计算 $\mathbf{S}_i$。为什么这不是一种好的想法？
3. 对于共享任意协方差矩阵 $\mathbf{S}$(7.15)式和 s^2 的情况，共享对角协方差矩阵(7.16)式的情况，为 $\mathbf{S}$ 推导 M 步公式。
4. 定义多元伯努利混合模型，其中输入是二元的，并推导 EM 公式。

7.11 参考文献

Alpaydin, E. 1998. "Soft Vector Quantization and the EM Algorithm." *Neural Networks* 11: 467 – 477.

Barrow, H. B. 1989. "Unsupervised Learning." *Neural Computation* 1: 295 – 311.

Bezdek, J. C., and N. R. Pal. 1995. "Two Soft Relatives of Learning Vector Quantization." *Neural Networks* 8: 729 – 743.

Bishop, C. M. 1999. "Latent Variable Models," In *Learning in Graphical Models*, ed. M. I. Jordan. 371 – 403. Cambridge, MA: The MIT Press.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of Royal Statistical Society* B 39: 1 – 38.

Gersho, A., and R. M. Gray. 1992. *Vector Quantization and Signal Compression*. Boston: Kluwer.

Ghahramani, Z., and G. E. Hinton. 1997. *The EM Algorithm for Mixtures of Factor Analyzers*. Technical Report CRG TR – 96 – 1, Department of Computer Science, University of Toronto.

Jain, A. K., and R. C. Dubes. 1988. *Algorithms for Clustering Data*. New York: Prentice Hall.

Jain, A. K., M. N. Murty, and P. J. Flynn. 1999. "Data Clustering: A Review." *ACM Computing Surveys* 31: 264 – 323.

McLachlan, G. J., and K. E. Basford. 1988. *Mixture Models: Inference and Applications to Clustering*. New York: Marcel Dekker.

McLachlan, G. J., and T. Krishnan. 1997. *The EM Algorithm and Extensions*. New York: Wiley.

Moerland, P. 1999. "A Comparison of Mixture Models for Density Estimation," In *International Conference on Artificial Neural Networks*, 25 – 30.

Ormoneit, D., and V. Tresp. 1996. "Improved Gaussian Mixture Density Estimates using Bayesian Penalty Terms and Network Averaging." In *Advances in Neural Information Processing Systems 8*, ed. D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, 542 – 548. Cambridge, MA: The MIT Press.

Redner, R. A., and H. F. Walker. 1984. "Mixture Densities, Maximum Likelihood and the EM Algorithm." *SI-*

AM Review 26: 195–239.

Rubin, D. B., and D. T. Thayer. 1982. "EM Algorithms for ML Factor Analysis." *Psychometrika 47*: 69–76.

Tipping, M. E., and C. M. Bishop. 1999. "Mixtures of Probabilistic Principal Component Analyzers." *Neural Computation* 11: 443–482.

Titerington, D. M., A. F. M. Smith, and E. E. Makov. 1985. *Statistical Analysis of Finite Mixture Distributions*. New York: Wiley.

Xu, L., and M. I. Jordan. 1996. "On Convergence Properties of the EM Algorithm for Gaussian Mixtures." *Neural Computation* 8: 129–151.

第8章 非参数方法

在前面的章节中，我们讨论了参数和半参数方法。在那里，我们假定数据取自一个形式已知的概率分布或混合分布。现在，我们将讨论非参数方法。当输入密度上不能做这样的假定时，可以使用非参数方法，并让数据自己说话。我们考虑密度估计、分类和回归的非参数方法，并讨论它们的时间和空间复杂度。

8.1 引言

在参数方法中，无论是密度估计、分类还是回归，我们都假设了一个在整个输入空间上有效的模型。例如，在回归中，当我们假定线性模型时，我们假定对于任何输入，输出都是输入的相同的线性函数。在分类中，当我们假定正态密度时，我们假定类的所有实例都取自这个相同的密度。参数方法的优点是，它把估计概率密度、判别式或回归函数问题归结为估计少量参数值。它的缺点是，假定并非总是成立的，并且不成立时可能导致很大的误差。

如果我们不能做这种假设并且不能使用参数模型时，一种可能的方法是使用如同我们在第7章所看到的半参数的混合模型，其中密度表示成几个参数模型的析取。在非参数估计(nonparametric estimation)中，我们只假定相似的输入具有相似的输出。这是一种合理的假设：世界是平稳的，并且无论是密度、判别式还是回归函数都缓慢地变化。相似的实例意味相似的事物。我们都爱我们的邻居，因为他们太像我们。

这样，我们的算法使用合适的距离度量，从训练集中找出相似的实例，并且由它们插值，得到正确的输出。不同的非参数方法在定义相似性或由相似的训练实例插值方法方面不同。在参数模型中，所有的训练实例都影响最终的全局估计。而在非参数情况下，不存在单个全局模型；需要时，局部模型被估计，它们只受邻近实例的影响。

在机器学习的文献中，非参数方法又称基于实例(instance-based)或基于记忆(memory-based)的学习算法，因为它们所做的是把训练实例存放在一个查找表中，并且由它们插值。这意味所有的训练实例都要存放，而存放所有训练实例需要$\mathcal{O}(N)$存储量。此外，给定一个输入，应当找出相似的训练实例，而找出它们需要$\mathcal{O}(N)$计算量。这种方法也称惰性(lazy)学习算法，因为不像急切(eager)的参数方法，当给定训练集时，它们并不计算模型，而是将模型的计算推迟到给定一个检验实例时才进行。对于参数学习方法，模型都相当简单，具有$\mathcal{O}(d)$或$\mathcal{O}(d^2)$量级个参数，并且一旦从训练集中计算出这些参数，我们保存模型并且在计算输出时就不再需要训练集了。通常，N比d(或d^2)大得多，而这种存储和计算量的增加是非参数方法的缺点。

我们从估计密度函数开始，并讨论它在分类上的应用。然后，我们将该方法推广到回归中。

8.2 非参数密度估计

与通常的密度估计一样，我们假设样本$\mathcal{X}=\{x^t\}_{t=1}^N$独立地从一个未知的概率密度$p(\cdot)$中

抽取。$\hat{p}(\cdot)$是$p(\cdot)$的估计。我们从单变量情况开始，其中x^t是标量，而稍后我们推广到多维情况。

154 累积分布函数$F(x)$在点x的非参数估计是小于或等于x的样本所占的比例

$$\hat{F}(x) = \frac{\#\{x^t \leq x\}}{N} \tag{8.1}$$

其中$\#\{x^t \leq x\}$表示其x^t小于或等于x的训练样本数。类似地，密度函数的非参数估计可以用下式计算

$$\hat{p}(x) = \frac{1}{h}\left[\frac{\#\{x^t \leq x+h\} - \#\{x^t \leq x\}}{N}\right] \tag{8.2}$$

其中h是区间长度，并且假定落入该区间中的实例x^t是“足够接近”的。本章提供的技术是一些变体，使用不同的启发式策略来确定邻近的实例和它们对估计的影响。

8.2.1 直方图估计

最古老、最流行的方法是直方图(histogram)。在直方图中，输入空间被划分成称作箱的相等区间。给定原点x_0和箱宽度h，箱是区间$[x_0+mh, x_0+(m+1)h)$(m是正整数或负整数)，而估计由下式给出

$$\hat{p}(x) = \frac{\#\{x^t \text{与} x \text{在相同的箱中}\}}{Nh} \tag{8.3}$$

在构造直方图时，我们必须选取原点和箱宽度。原点的选取影响靠近箱边界的估计，但影响估计的主要是箱宽度：使用小箱，估计是尖峰的，而使用大箱，估计较光滑(参见图8-1)。如果没有实例落入箱中，则估计为0，并且在箱边界处不连续。直方图的优点是：一旦计算和存放了箱估计，我们就不再需要保留训练集。

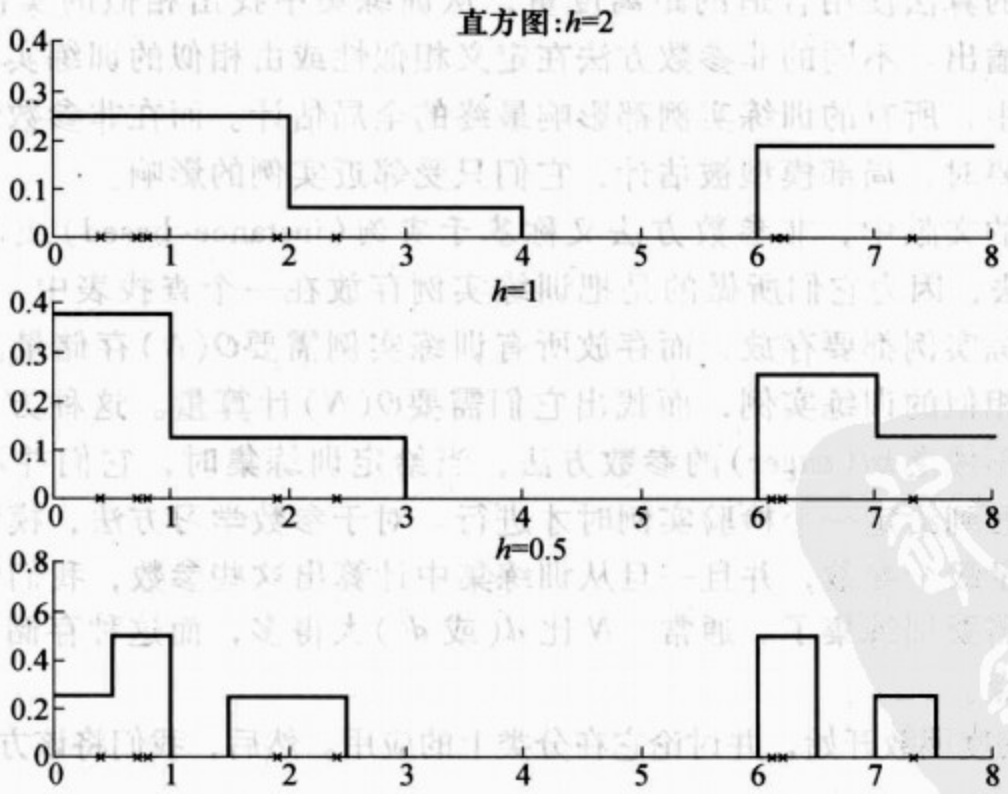


图8-1 各种箱长度的直方图。“×”表示数据点

质朴估计法(naive estimator)(Silverman 1986)使得我们不必设置原点。它定义为

$$\hat{p}(x) = \frac{\#\{x - h < x^t \leqslant x + h\}}{2Nh} \tag{8.4}$$

并且它等于 x 总是落在宽度为 $2h$ 的箱中心的直方图估计(参见图 8-2)。该估计还可以表示为

$$\hat{p}(x) = \frac{1}{Nh}\sum_{t=1}^{N} w\left(\frac{x - x^t}{h}\right) \tag{8.5}$$

其中权重函数定义为

$$w(u) = \begin{cases} \frac{1}{2} & \text{如果 } |u| < 1 \\ 0 & \text{否则} \end{cases}$$

这就好像每个 x^t 都有一个围绕它的大小为 $2h$ 的、对称的影响区域，并且对落入该区域的 x 贡献 1/2。于是，非参数估计恰为其区域包含 x 的 x^t 的影响之和。因为这种影响区域是“硬的”(0 或 1/2)，所以估计不是连续函数并在 $x^t \pm h$ 有跳跃。

155
≀
156

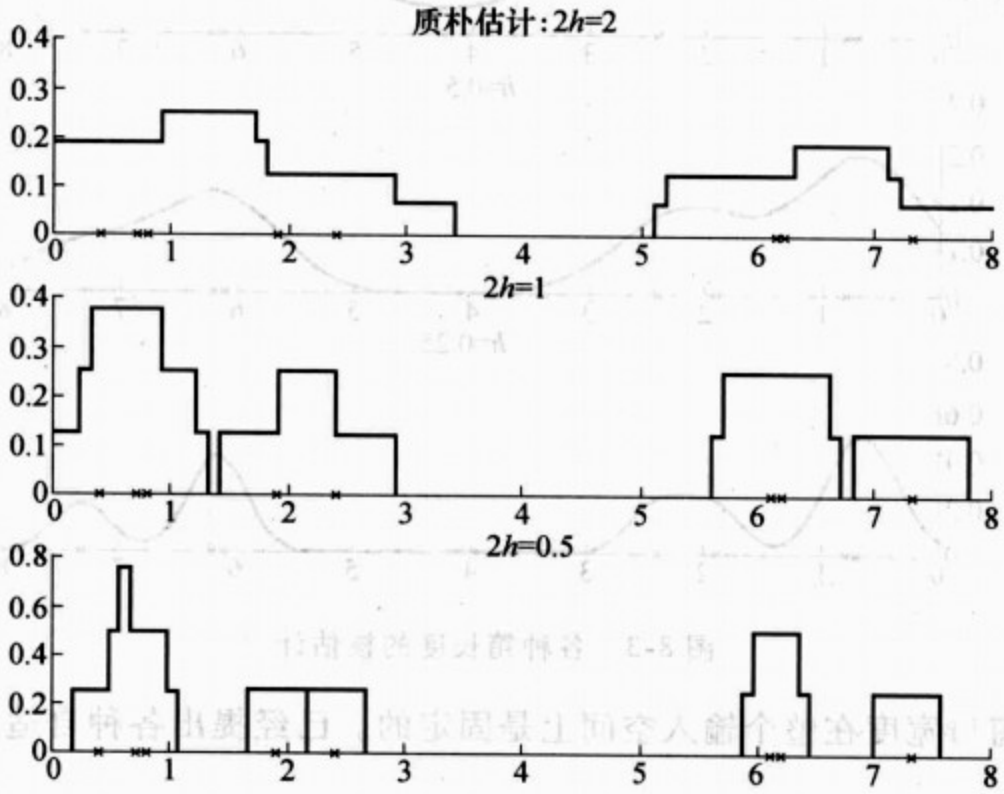


图 8-2 各种箱长度的质朴估计

8.2.2 核估计

为了得到光滑的估计，我们使用一个光滑的权重函数，称作核函数(kernel function)。最流行的是高斯核：

$$K(u) = \frac{1}{\sqrt{2\pi}}\exp\left[-\frac{u^2}{2}\right] \tag{8.6}$$

核估计(kernel estimator)又称 *Parzen* 窗口(Parzen windows)，定义为

$$\hat{p}(x) = \frac{1}{Nh}\sum_{t=1}^{N} K\left(\frac{x - x^t}{h}\right) \tag{8.7}$$

核函数 $K(\cdot)$ 决定影响的形状，而窗口宽度 h 决定影响的宽度。就像质朴估计是“箱”

的和一样，核估计是“凸块”的和。所有的 x^t 都对 x 上的估计具有影响，并且其影响随 $|x-x^t|$ 的增加而平滑地减小。

为了简化计算，如果 $|x-x^t|>3h$，则 $K(\cdot)$ 可以取 0。还可以使用其他容易计算的核函
157 数，只要 $K(u)$ 对 $u=0$ 取最大值，并且随 $|u|$ 增加而对称地减少。

当 h 很小时，每个训练实例在一个小区域都具有较大影响，而在较远的点上没有影响。当 h 较大时，有更多的核重叠，而我们得到较光滑的估计(参见图 8-3)。如果 $K(\cdot)$ 处处非负并且积分为 1，即如果它是合法的密度函数，则 $\hat{p}(\cdot)$ 也是。此外，$\hat{p}(\cdot)$ 将继承核函数 $K(\cdot)$ 的连续性和可微性。例如，如果 $K(\cdot)$ 是高斯函数，则 $\hat{p}(\cdot)$ 将是光滑的并且具有所有导数。

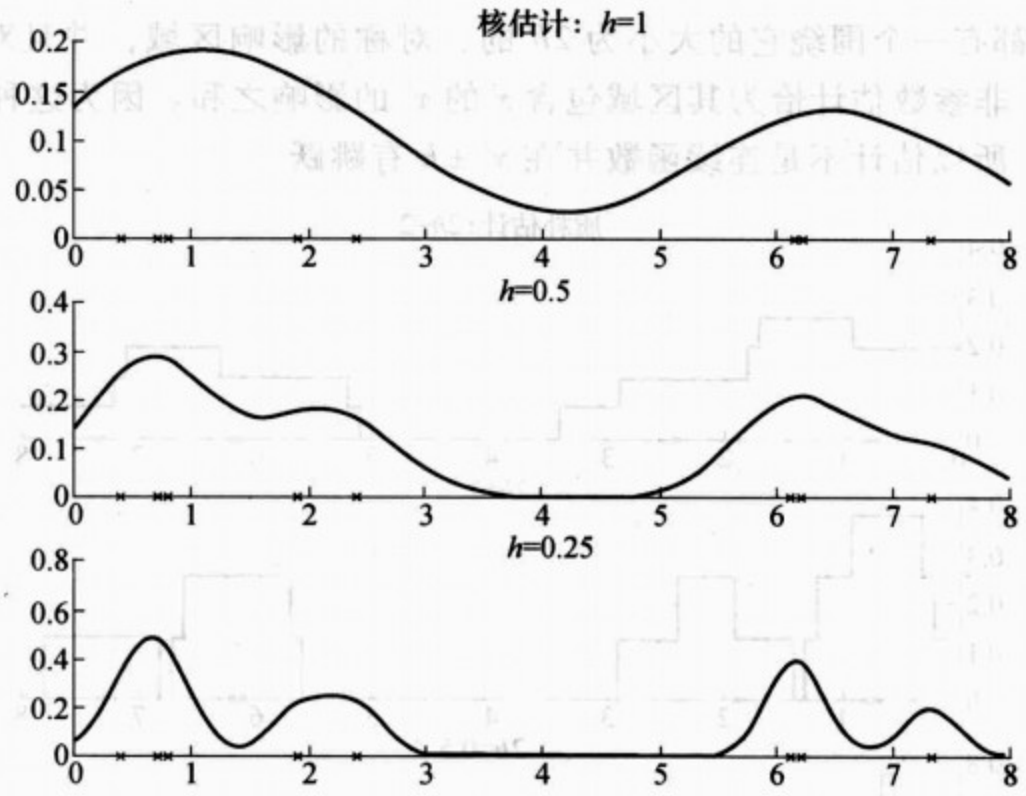


图 8-3 各种箱长度的核估计

一个问题是窗口宽度在整个输入空间上是固定的。已经提出各种自适应方法将 h 看作 x 周围密度的函数。

8.2.3 k-最近邻估计

估计的最近邻方法调整光滑量使之适应数据的局部密度。光滑度被所考虑的近邻数 k 控
158 制。近邻数 k 远小于样本规模 N。我们定义 a 和 b 之间的距离，例如为 $|a-b|$，并对每个 x，我们定义

$$d_1(x) \leqslant d_2(x) \leqslant \cdots \leqslant d_N(x)$$

为从 x 到样本中的点按递减序排列的距离：$d_1(x)$ 是最近的样本的距离，$d_2(x)$ 是次近样本的距离，如此下去。如果 x^t 是数据点，则我们定义 $d_1(x)=\min_t |x-x^t|$，并且如果 i 是最近样本的下标，即 $i=\arg\min_t |x-x^t|$，则 $d_2(x)=\min_{j\neq i} |x-x^j|$，如此下去。

k-最近邻(k-nearest neighbor，k-nn)密度估计为

$$\hat{p}(x)=\frac{1}{2Nd_k(x)} \tag{8.8}$$

这就像 $h=d_k(x)$ 的质朴估计，不同之处是我们不是固定 h 并检查多少样本落入箱中，而是固定落入箱中的观测数 k，并计算箱的大小。密度高的地方箱较小，而密度低的地方箱较大(参见图 8-4)。

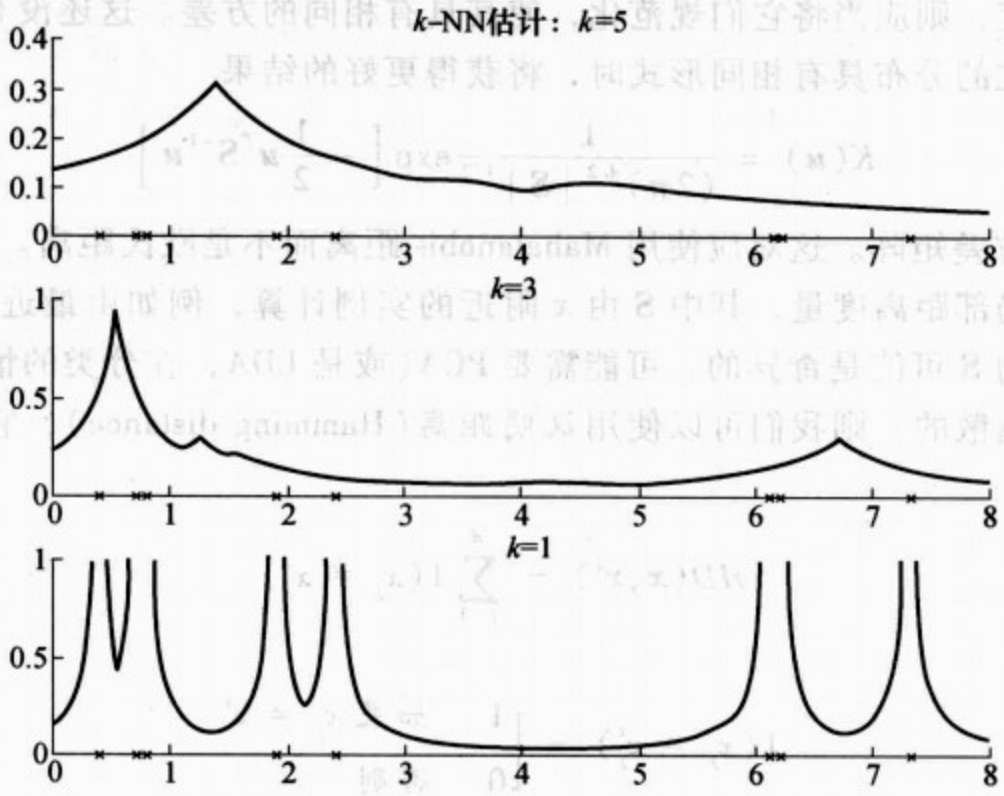


图 8-4 各种 k 值的 k-最近邻估计

k-nn 估计不是连续的；它的导数在所有的 $\frac{1}{2}(x^{(j)}+x^{(j+k)})$ 上不具有连续性，其中 $x^{(j)}$ 是样本的顺序统计量。k-nn 不是概率密度函数，因为它的积分为 ∞，而不是 1。

为了得到更光滑的估计，我们可以使用其影响随距离增加而减小的核函数

$$\hat{p}(x)=\frac{1}{Nd_k(x)}\sum_{t=1}^{N}K\left(\frac{x-x^t}{d_k(x)}\right) \tag{8.9}$$

这就像具有自适应光滑参数 $h=d_k(x)$ 的核估计。通常，$K(\cdot)$ 取高斯核。

8.3 到多变元数据的推广

给定 d-维观测的样本 $\mathcal{X}=\{\boldsymbol{x}^t\}_{t=1}^{N}$，多元核密度估计为

$$\hat{p}(\boldsymbol{x})=\frac{1}{Nh^d}\sum_{t=1}^{N}K\left(\frac{\boldsymbol{x}-\boldsymbol{x}^t}{h}\right) \tag{8.10}$$

满足必要条件

$$\int_{\Re^d}K(\boldsymbol{x})d\boldsymbol{x}=1$$

一个显然的候选是多元高斯核：

$$K(\boldsymbol{u})=\left(\frac{1}{\sqrt{2\pi}}\right)^d\exp\left[-\frac{\|\boldsymbol{u}\|^2}{2}\right] \tag{8.11}$$

然而，由于维灾难(curse of dimensionality)，在高维空间使用非参数估计时需要小心：令 $\boldsymbol{x}$ 是 8 维的，我们使用每维 10 个箱的直方图，则有 10^8 个箱。除非我们有大量数据，否

则大部分箱为空，并且那里的估计为0。在高维空间，“近邻”概念也变得模糊不清，因此在选择 h 时需要小心。

例如，(8.11)式中欧几里德范数的使用意味核在所有维上都具有相等的尺度。如果输入具有不同的尺度，则应当将它们规范化，使其具有相同的方差。这还没有考虑相关性，并且当核函数与潜在的分布具有相同形式时，将获得更好的结果

159
ı
160

$$K(\boldsymbol{u}) = \frac{1}{(2\pi)^{d/2}|\mathbf{S}|^{1/2}}\exp\left[-\frac{1}{2}\boldsymbol{u}^T\mathbf{S}^{-1}\boldsymbol{u}\right] \tag{8.12}$$

其中 $\mathbf{S}$ 是样本协方差矩阵。这对应使用 Mahalanobis 距离而不是欧氏距离。

也可以使用局部距离度量，其中 $\mathbf{S}$ 由 $\boldsymbol{x}$ 附近的实例计算，例如由最近的 k 个实例计算。注意，局部计算的 $\mathbf{S}$ 可能是奇异的，可能需要 PCA(或是 LDA，在分类的情况中)。

如果输入是离散的，则我们可以使用汉明距离(Hamming distance)，它对不匹配的属性计数

$$HD(\boldsymbol{x},\boldsymbol{x}^t) = \sum_{j=1}^{d} 1(x_j \neq x_j^t) \tag{8.13}$$

其中

$$1(x_j \neq x_j^t) = \begin{cases} 1 & \text{如果 } x_j \neq x_j^t \\ 0 & \text{否则} \end{cases}$$

然后，在使用 $\|\boldsymbol{x}-\boldsymbol{x}^t\|$ 或 $(\boldsymbol{x}-\boldsymbol{x}^t)^T\mathbf{S}^{-1}(\boldsymbol{x}-\boldsymbol{x}^t)$ 的地方使用 $HD(\boldsymbol{x},\boldsymbol{x}^t)$ 进行核估计或找出 k 个最近邻。

8.4 非参数分类

当用于分类时，我们使用非参数方法估计类条件密度 $p(\boldsymbol{x}|C_i)$。类条件密度的核估计由下式给出

$$\hat{p}(\boldsymbol{x}|C_i) = \frac{1}{N_i h^d}\sum_{t=1}^{N} K\left(\frac{\boldsymbol{x}-\boldsymbol{x}^t}{h}\right) r_i^t \tag{8.14}$$

其中如果 $\boldsymbol{x}^t \in C_i$，则 r_i^t 为1，否则 r_i^t 为0。N_i 是属于 C_i 的标记实例数；$N_i = \Sigma_t r_i^t$。先验密度的 MLE 是 $\hat{P}(C_i) = N_i/N$。于是，判别式可以表示为

$$\begin{aligned} g_i(\boldsymbol{x}) &= \hat{p}(\boldsymbol{x}|C_i)\hat{P}(C_i) \\ &= \frac{1}{Nh^d}\sum_{t=1}^{N} K\left(\frac{\boldsymbol{x}-\boldsymbol{x}^t}{h}\right) r_i^t \end{aligned} \tag{8.15}$$

并且 $\boldsymbol{x}$ 被指派到判别式取最大值的类。公共因子 $1/(Nh^d)$ 可以忽略。这样，每个训练实例都
161 为它的类投票，而对其他类没有影响；投票的权重由核函数 $K(\cdot)$ 给定，通常赋予更近的实例更高的权重。

对于 k-nn 估计的特殊情形，我们有

$$\hat{p}(\boldsymbol{x}|C_i) = \frac{k_i}{N_i V^k(\boldsymbol{x})} \tag{8.16}$$

其中 k_i 是 k 个最近邻中属于 C_i 的近邻数，而 $V^k(\boldsymbol{x})$ 是中心在 $\boldsymbol{x}$，半径为 $r = \|\boldsymbol{x}-\boldsymbol{x}_{(k)}\|$ 的

d-维超球的体积，这里 $\boldsymbol{x}_{(k)}$ 是(所有类 $\boldsymbol{x}$ 的近邻中)第 k 个距离 $\boldsymbol{x}$ 最近的观测：$V^k = r^d c_d$，c_d 是 d 维单位球的体积。例如，$c_1 = 2$，$c_2 = \pi$，$c_3 = 4\pi/3$，如此等等。于是

$$\hat{P}(C_i \mid \boldsymbol{x}) = \frac{\hat{p}(\boldsymbol{x} \mid C_i)\hat{P}(C_i)}{\hat{p}(\boldsymbol{x})} = \frac{k_i}{k} \tag{8.17}$$

k-nn 分类法(k-nn classifier)将输入指派到输入的 k 个最近邻中具有最多实例的类。所有的近邻都有相同的投票权，并且选取 k 个近邻中具有最多投票者的类。平局随意打破或用加权投票。通常，k 取奇数，以减少平局：难以区分的情况一般出现在两个相邻的类之间。

再则，欧氏距离的使用对应假定不相关的输入具有相同的方差。如果不是这种情况，应当选择合适的度量。一个例子是判别式自适应最近邻(discriminant adaptive nearest neighbor)(Hastie 和 Tibshirani 1996)，那里局部地估计将类分离的最佳距离。

k-nn 的一种特殊情况是最近邻分类(nearest neighbor classifier)，其中 $k = 1$，并且输入被指派到最近的模式所在的类。这将空间划分成 Voronoi 图[⊖](Voronoi tesselation)形式(参见图 8-5)。

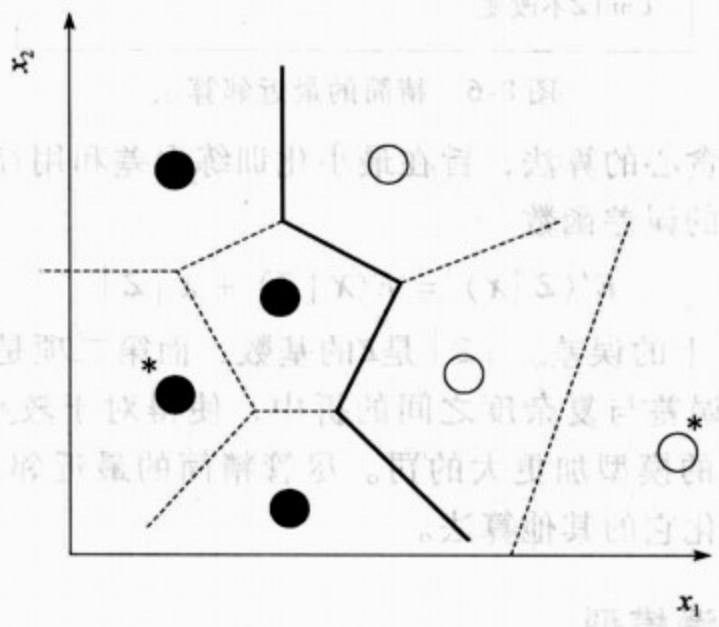


图 8-5 虚线是 Voronoi 图，而实线是判别式。在精简的最近邻中，可以删除那些不参与定义判别式的实例(用“*”标记)而不增加训练误差

8.5 精简的最近邻

非参数方法的时间和空间复杂度与训练集的大小成正比。已经提出了一些精简方法，以减少存放的实例数而不降低性能。其基本思想是选择 $\mathcal{X}$ 的最小子集 $\mathcal{Z}$ 使得用 $\mathcal{Z}$ 替代 $\mathcal{X}$ 时，误差不增加(Dasarathy 1991)。

最著名和最早的方法是精简的最近邻(condensed nearest neighbor)，那里使用 1-nn 作为分类的非参数估计(Hart 1968)。1-nn 以分段线形的方式逼近判别式，并且只需要保存定义

⊖ Voronoi tesselation 又称 Voronoi diagram，是 Georgy Voronoi 提出的。它由一组由连接两个相邻点线段的垂直平分线组成的多边形组成。——译者注

判别式的实例。类区域内部的实例不必作为它的同一类的最近邻存放，并且它的缺失不会导致(训练集上的)任何错误(图8-5)。这样的子集称作相容子集，并且我们希望找出最小的相容子集。

Hart提出了一种发现$\mathcal{Z}$的贪心算法(图8-6)：该算法从空集$\mathcal{Z}$开始，以随机次序逐个扫描$\mathcal{X}$中的实例，并检查它们是否能够被1-nn用已经在$\mathcal{Z}$中的实例正确地分类。如果一个实例被错误分类，则将它添加到$\mathcal{Z}$中；如果它被正确分类，则$\mathcal{Z}$不变。应当扫描数据集多遍，直到没有实例再添加到$\mathcal{Z}$中。该算法进行局部搜索，并且依赖于看到训练实例的次序，可能找出不同的子集，每个子集在确认数据上具有不同的准确率。因此，不能保证找到最小的相容子集。找出最小相容子集是NP-完全问题(Wilfong 1992)。

162
~
163

$\mathcal{Z} \leftarrow \emptyset$
Repeat
For 所有的 $x \in \mathcal{X}$(以随机次序)
找出 $x' \in \mathcal{Z}$ 使得 $\|x - x'\| = \min_{x^j \in \mathcal{Z}} \|x - x^j\|$
If $\text{class}(x) \neq \text{class}(x')$ 将 x 添加到 $\mathcal{Z}$ 中
Until $\mathcal{Z}$ 不改变

图8-6 精简的最近邻算法

精简的最近邻是一种贪心的算法，旨在最小化训练误差和用存放的子集规模度量的复杂度。我们可以写一个增广的误差函数

$$E'(\mathcal{Z} \mid \mathcal{X}) = E(\mathcal{X} \mid \mathcal{Z}) + \lambda \mid \mathcal{Z} \mid \tag{8.18}$$

其中$E(\mathcal{X} \mid \mathcal{Z})$是存放$\mathcal{Z}$在$\mathcal{X}$上的误差。$\mid \mathcal{Z} \mid$是$\mathcal{Z}$的基数，而第二项是对复杂度加罚。与所有的正则化方案一样，λ体现误差与复杂度之间的折中，使得对于较小的λ，误差变得更重要，并且随着λ增大，对复杂的模型加更大的罚。尽管精简的最近邻是一种最小化(8.18)式的方法，但是还可以设计优化它的其他算法。

8.6 非参数回归：光滑模型

在回归中，给定训练集$\mathcal{X} = \{x^t, r^t\}$，其中$r^t \in \Re$，我们假定

$$r^t = g(x^t) + \varepsilon$$

在参数回归，我们假定某次多项式，并计算它的系数，最小化训练集上误差的平方和。当不能假定这种多项式时，使用非参数回归；我们只假定相近的x具有相近的$g(x)$值。与非参数密度估计一样，给定x，我们的方法是找到x的邻域，并求邻域中r的平均值，计算$\hat{g}(x)$。非参数回归估计子又称光滑子(smoother)，而该估计称作光滑(Härdle 1990)。类似于
164 密度估计，存在各种定义邻域和在邻域中取平均值的方法。我们对一元变量x讨论这些方法；与密度估计一样，使用多元核函数，可以用直截了当的方式把它们推广到多元情况中。

8.6.1 移动均值光滑

如果我们像在直方图中那样，定义一个原点和箱宽度并在箱中求r的平均值，则我们得到回归图(regressogram)(参见图8-7)。

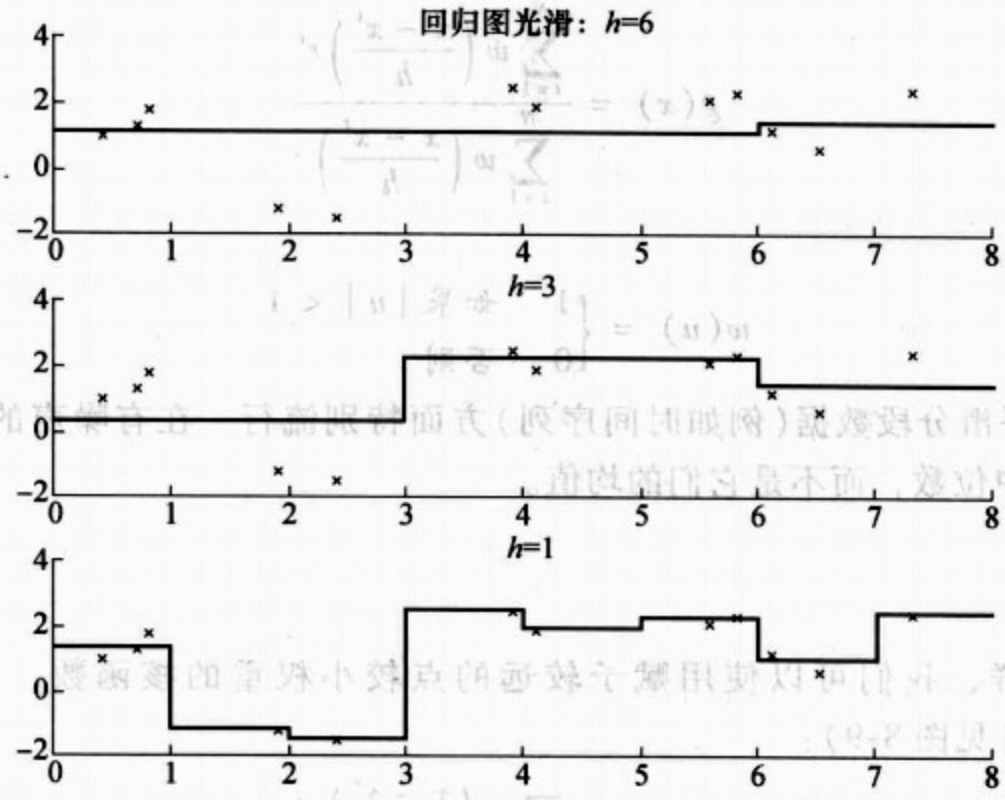


图 8-7　各种箱长度的回归图。“×”表示数据点

$$\hat{g}(x) = \frac{\sum_{t=1}^{N} b(x,x^t) r^t}{\sum_{t=1}^{N} b(x,x^t)} \tag{8.19}$$

其中

$$b(x,x^t) = \begin{cases} 1 & \text{如果 } x^t \text{ 与 } x \text{ 在同一个箱中} \\ 0 & \text{否则} \end{cases}$$

由于需要固定原点，箱边界上的不连续是令人烦恼的。像质朴估计一样，在移动均值光滑(running mean smoother)中，我们在 x 周围定义一个对称的箱并在那里取平均值(图 8-8)。

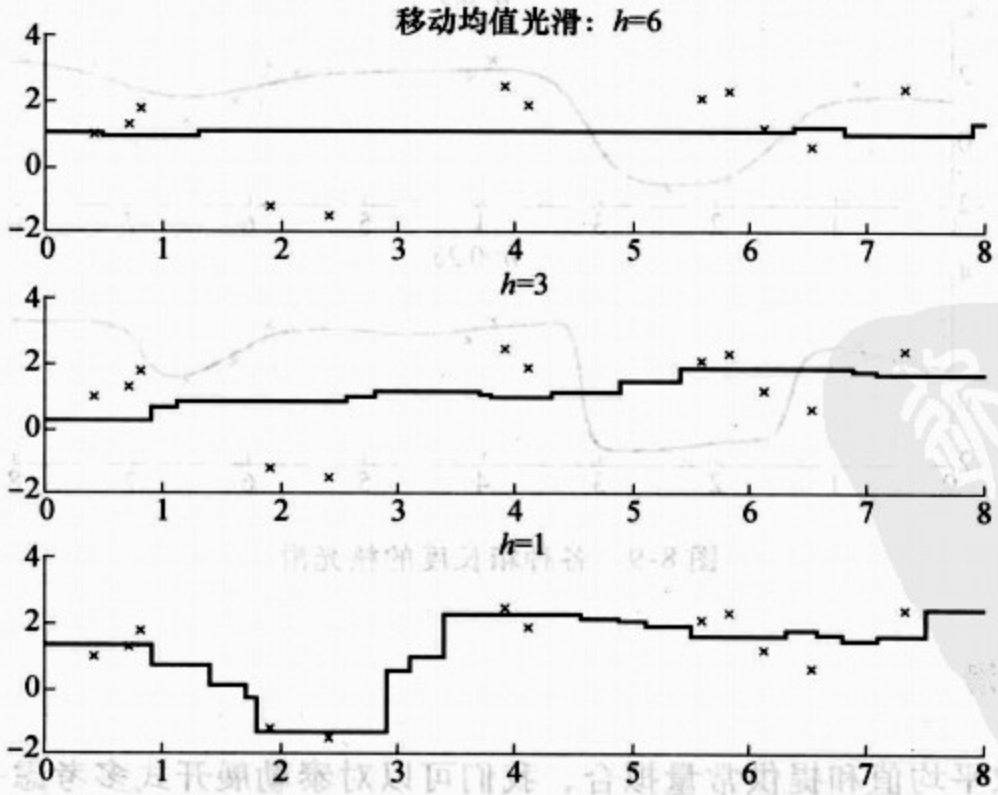


图 8-8　各种箱长度的移动均值光滑

$$\hat{g}(x) = \frac{\sum_{t=1}^{N} w\left(\frac{x - x^t}{h}\right) r^t}{\sum_{t=1}^{N} w\left(\frac{x - x^t}{h}\right)} \tag{8.20}$$

其中

$$w(u) = \begin{cases} 1 & \text{如果 } |u| < 1 \\ 0 & \text{否则} \end{cases}$$

这种方法在平滑分段数据(例如时间序列)方面特别流行。在有噪声的应用中，我们可以使用箱中 r^t 的中位数，而不是它们的均值。

8.6.2 核光滑

与核估计一样，我们可以使用赋予较远的点较小权重的核函数，并且得到核光滑(kernel smoother)(见图 8-9)：

165
~
166

$$\hat{g}(x) = \frac{\sum_{t} K\left(\frac{x - x^t}{h}\right) r^t}{\sum_{t} K\left(\frac{x - x^t}{h}\right)} \tag{8.21}$$

通常使用高斯核 $K(\cdot)$。替换固定 h，我们可以固定近邻数 k，使得估计自动适应 x 周围的密度，并且得到 k-nn 光滑(k-nn smoother)。

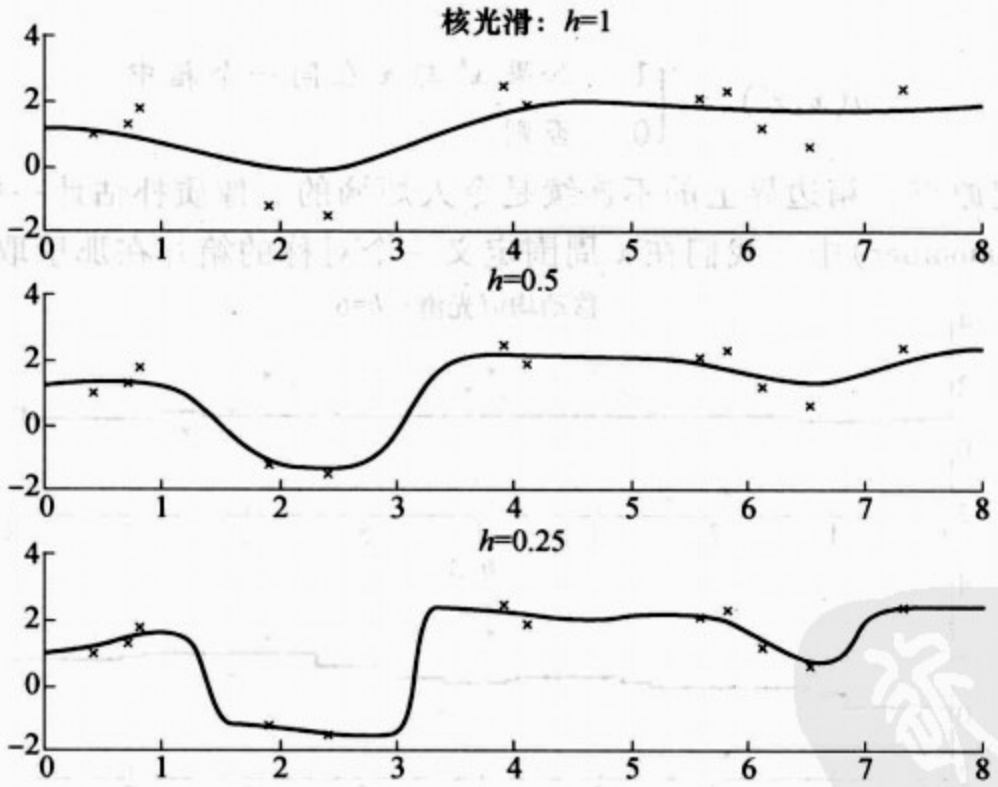


图 8-9 各种箱长度的核光滑

8.6.3 移动线光滑

替代在点上取平均值和提供常量拟合，我们可以对泰勒展开式多考虑一项并计算直线拟合。在移动线光滑(running line smoother)中，我们可以使用邻域(被 h 或 k 定义)中的数据

点，并拟合一个局部回归(见图 8-10)。

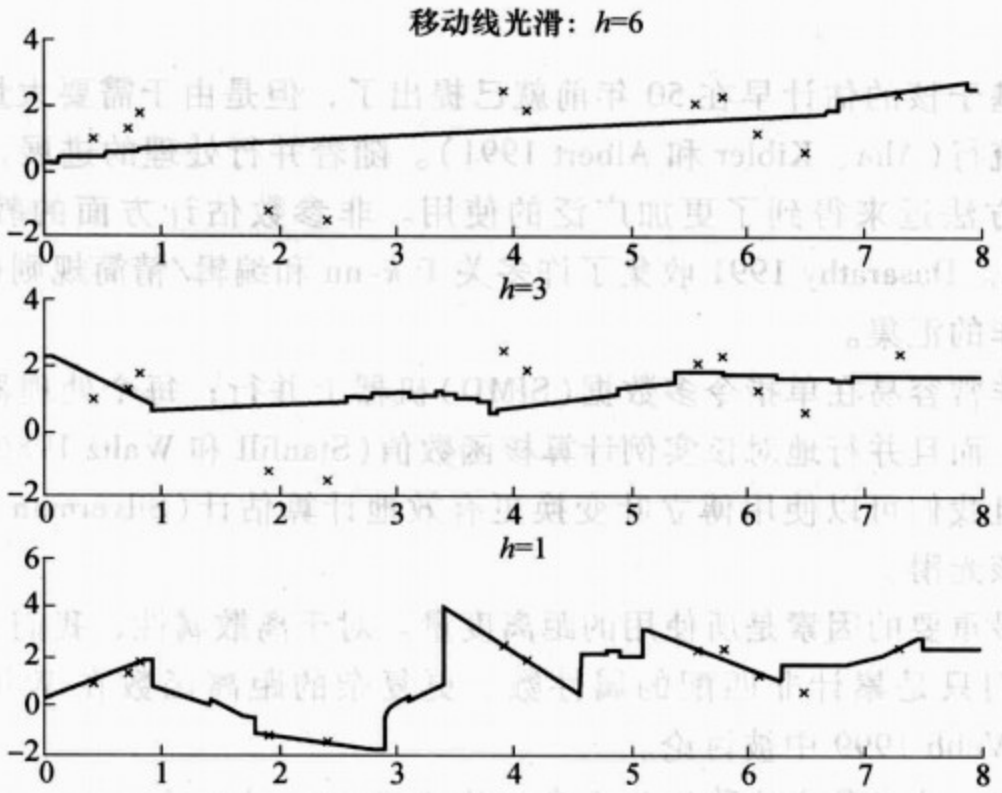


图 8-10 各种箱长度的移动线光滑

在局部加权移动线光滑(locally weighted running line smoother，称作 loess[⊖])中，我们使用核加权使得较远的点对误差具有较小影响，而不是使用邻域的僵硬定义。

8.7 如何选择光滑参数

在非参数方法中，对于密度估计或回归，关键的参数是光滑参数，如箱宽度或核扩展中的 h，或近邻数 k。目标是使得估计的不稳定性比数据点小。正如我们在前面已经讨论的，数据中的易变性的一个根源是噪声，其他根源是未知的潜在函数。我们应当光滑得恰好足以克服噪声——不少也不多。使用太大的 h 或 k，许多实例都对点上的估计做出了贡献，我们也光滑掉了源于函数的变化，存在过分光滑。使用太小的 h 或 k，单个实例具有很大影响，我们甚至没有光滑掉噪声，存在光滑不足。换句话说，小 h 或 k 导致小偏倚但大方差。大 h 或 k 降低方差但增加偏倚。Geman、Bienenstock 和 Doursat(1992)讨论了非参数估计的偏倚和方差。

167
~
168

这一要求明确地表示在正如光滑样条(smoothing splines)中所使用的正则化函数中

$$\sum_t [r^t - \hat{g}(x^t)]^2 + \lambda \int_a^b [\hat{g}''(x)]^2 dx \tag{8.22}$$

第一项是拟合的误差。$[a, b]$是输入区间；$\hat{g}''(\cdot)$是估计函数 $\hat{g}(\cdot)$的曲率(curvature)，因此度量变化。这样，第二项处罚快速变化的估计。λ 权衡变化和误差。例如，使用大 λ，我们得到更光滑的估计。

交叉确认用来调整 h，k 或 λ。在密度估计中，我们选择最大化确认集上似然的参数值。在监督环境下，在训练集上试验一系列候选，选取最小化确认集上误差的参数值。

⊖ loess 意为局部回归。——译者注

8.8 注释

k-最近邻和基于核的估计早在50年前就已提出了，但是由于需要大量存储和计算，该方法直到最近才流行(Aha、Kibler和Albert 1991)。随着并行处理的进展，存储和计算价格逐渐降低，这些方法近来得到了更加广泛的使用。非参数估计方面的教科书是Silverman 1986和Scott 1992。Dasarathy 1991收集了许多关于k-nn和编辑/精简规则的文章。Aha 1997是更加近期的工作的汇集。

非参数方法非常容易在单指令多数据(SIMD)机器上并行；每个处理器在其局部存储器中存放一个实例，而且并行地对该实例计算核函数值(Stanfill和Waltz 1986)。乘以核函数可以看作卷积，并且我们可以使用傅立叶变换更有效地计算估计(Silverman 1986)。已经证明样条光滑等价于核光滑。

非参数估计最重要的因素是所使用的距离度量。对于离散属性，我们可以简单地使用汉明距离，那里我们只是累计非匹配的属性数。更复杂的距离函数在Wettschereck，Aha和Mohri 1997以及Webb 1999中被讨论。

在人工智能中，非参数方法称作*基于案例的推理*(case-based reasoning)。通过对已知的
169 类似旧“案例”插值找到输出。这也允许知识提取：给定的输出可以用列举这些类似的旧案例证明其合理性。

由于其简单性，k-nn是最广泛使用的非参数分类方法，并且在各种实践应用中相当成功。已经证明(Cover和Hart 1967；又见Duda、Hart和Stork 2001)：在大样本中，当$N\to\infty$时，最近邻($k=1$)的风险不超过贝叶斯风险(我们能够得到的最好结果)的两倍，并且从这方面来讲，可以说“在被分类的无限样本集中，一般的可用信息包含在最近邻中”(Cover和Hart 1967)。对于k-nn，业已证明，随着k趋向于无穷大，其风险逼近贝叶斯风险。

非参数回归在Härdle 1990中详细讨论。Hastie和Tibshirani(1990)讨论了光滑模型并提出了加法模型(additive model)，其中多元函数被表示成一元估计的和。局部加权回归在Atkeson，Moore和Schaal 1997中讨论。这些模型与我们将在第12章讨论的径向基函数和混合专家模型很相似。

8.9 习题

1. 证明(8.17)式。
2. 如果$k>1$，精简最近邻会怎么样？
3. 在回归图中，替代箱中取平均值并做常量拟合，我们可以使用落入箱中的实例并做线性拟合(见图8-11)。写出代码并与回归图做适当比较。
4. 为8.6.3节讨论的loess写出误差函数。
5. 提出一个移动均值估计的增量版本，像精简最近邻一样，它只在必要时存放实例。
6. 将核光滑推广到多元数据。

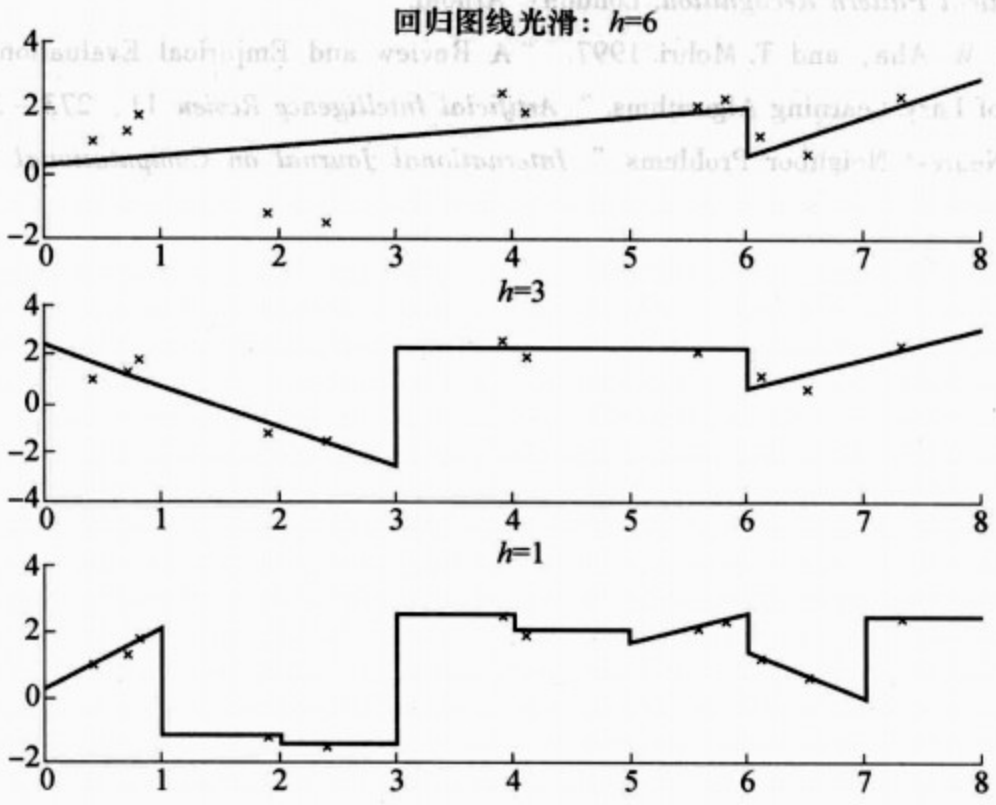


图 8-11 对于各种箱长度，使用线性拟合的回归图

8.10 参考文献

Aha, D. W., ed. 1997. Special Issue on Lazy Learning, *Artificial Intelligence Review* 11: issues 1 – 5.

Aha, D. W., D. Kibler, and M. K. Albert. 1991. "Instance- Based Learning Algorithm." *Machine Learning* 6: 37 – 66.

Atkeson, C. G., A. W. Moore, and S. Schaal. 1997. "Locally Weighted Learning." *Artificial Intelligence Review* 11: 11 – 73.

Cover, T. M., and P. E. Hart. 1967. "Nearest Neighbor Pattern Classification." *IEEE Transactions on Information Theory* 13: 21 – 27.

Dasarathy, B. V. 1991. *Nearest Neighbor Norms: NN Pattern Classification Techniques.* Los Alamitos, CA: IEEE Computer Society Press.

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, 2nd ed. New York: Wiley.

Geman, S., E. Bienenstock, and R. Doursat. 1992. "Neural Networks and the Bias/Variance Dilemma." *Neural Computation* 4: 1 – 58.

Härdle, W. 1990. *Applied Nonparametric Regression.* Cambridge, UK: Cambridge University Press.

Hart, P. E. 1968. "The Condensed Nearest Neighbor Rule." *IEEE Transactions on Information Theory* 14: 515 – 516.

Hastie, T. J., and R. J. Tibshirani. 1990. *Generalized Additive Models.* London: Chapman and Hall.

Hastie, T. J., and R. J. Tibshirani. 1996. "Discriminant Adaptive Nearest Neighbor Classification." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18: 607 – 616.

Scott, D. W. 1992. *Multivariate Density Estimation.* New York: Wiley.

Silverman, B. W. 1986. *Density Estimation in Statistics and Data Analysis.* London: Chapman and Hall.

Stanfill, C. and D. Waltz. 1986. "Toward Memory- Based Reasoning." *Communications of the ACM* 29: 1213 – 1228.

170
≀
171

Webb, A. 1999. *Statistical Pattern Recognition*. London: Arnold.

Wettschereck, D., D. W. Aha, and T. Mohri. 1997. "A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms." *Artificial Intelligence Review* 11: 273–314.

Wilfong, G. 1992. "Nearest Neighbor Problems." *International Journal on Computational Geometry and Applications* 2: 383–416.

172

第9章 决 策 树

决策树是一种实现分治策略的层次数据结构。它是一种有效的非参数学习方法，可以用于分类和回归。本章，我们讨论由给定的标记的训练样本构造决策树的学习算法，以及如何将决策树转换成容易理解的简单规则的方法。

9.1 引言

对于参数估计，我们定义整个输入空间上的模型，并使用所有的训练数据学习它的参数。然后，对任意的检验输入，使用相同的模型和参数。对于非参数估计，我们把输入空间划分成被诸如欧几里德范数这样的距离度量定义的局部区域，并对每个输入使用由该区域的训练数据计算得到的对应的局部模型。在非参数模型中，给定一个输入，识别定义局部模型的局部数据的开销很大，需要计算从给定的输入到所有训练实例的距离。其计算复杂度为$\mathcal{O}(N)$。

决策树(decision tree)是一种用于监督学习的层次模型，由此，局部区域通过少数几步递归分裂确定。决策树由一些内部决策节点和终端树叶组成(参见图9-1)。每个决策节点(decision node)m实现一个具有离散输出的测试函数$f_m(\boldsymbol{x})$，标记分支。给定一个输入，在每个节点应用一个测试，并根据测试的输出确定一个分支。这一过程从根节点开始，并递归地重复，直至到达一个树叶节点(leaf node)。这时，该树叶中的值形成输出。

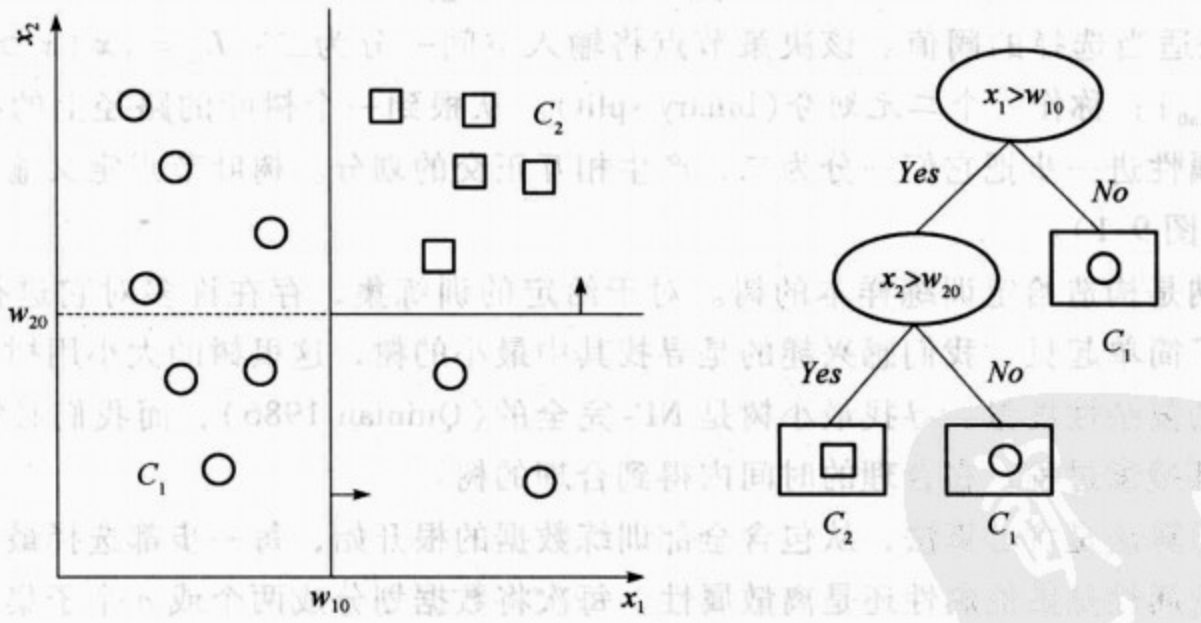


图9-1 数据集和对应的决策树。椭圆形节点是决策节点，而矩形节点是树叶节点。单变量的决策节点沿着一个轴划分，并且相继的划分相互正交。第一次划分之后，$\{\boldsymbol{x} \mid x_1 < w_{10}\}$已是纯的，因此不需要再划分

每个$f_m(\boldsymbol{x})$定义了一个d-维输入空间中的判别式，将空间划分成较小区域。在从根节点

沿一条路径向下时，这些较小的区域被进一步划分。$f_m(\cdot)$是一个简单函数，而作为树写下时，复杂的函数被分解成一系列简单决策。不同的决策树方法对$f_m(\cdot)$假设不同的模型，而模型类确定了判别式的形状和区域的形状。每个树叶节点有一个输出标号。对于分类，该标号是类代码；而对于回归，它是一个数值。一个树叶节点定义了输入空间的一个局部区域，落入该区域的实例具有相同的输出。区域的边界被从树根到该树叶的路径上的内部节点中的判别式定义。

173
~
174

决策的层次安排使得涵盖输入的区域可以快速确定。例如，如果决策是二元的，则在最好情况下每个决策去掉一半实例。如果有b个区域，则在最好情况下可以通过$\log_2 b$次决策找到正确的区域。决策树的另一个优点是可解释性：正如稍后我们将看到的，可以把决策树转换成一组容易理解的*IF-THEN*规则。因此，决策树非常流行，并且常常比更准确但是不太好解释的方法更可取。

我们从一个决策节点只使用一个输入变量的单变量树开始，考察如何为分类和回归构造这样的树。稍后，我们将这种方法推广到一个内部节点可以使用所有输入的多变量树。

9.2 单变量树

在单变量树(univariate tree)中，每个内部节点中的测试只使用一个输入维。如果所使用的输入维x_j是离散的，取n个可能的值之一，则该决策节点检查x_j的值，并取相应的分支，实现一个n路划分。例如，如果属性是颜色，具有可能的值{红，蓝，绿}，则该属性上的节点具有三个分支，每个对应该属性的三个可能值中的一个。

决策节点具有离散分支，而数值输入应当离散化。如果x_j是数值的(有序的)，则测试是比较

$$f_m(\boldsymbol{x}): x_j \geqslant w_{m0} \tag{9.1}$$

其中w_{m0}是适当选择的阈值。该决策节点将输入空间一分为二：$L_m = \{\boldsymbol{x} \mid x_j \geqslant w_{m0}\}$和$R_m = \{\boldsymbol{x} \mid x_j < w_{m0}\}$；称作一个二元划分(binary split)。从根到一个树叶的路径上的相继决策节点使用其他属性进一步把它们一分为二，产生相互正交的划分。树叶节点定义输入空间中的超矩形(参见图9-1)。

树归纳是构造给定训练样本的树。对于给定的训练集，存在许多对它进行无错编码的树，而为了简单起见，我们感兴趣的是寻找其中最小的树，这里树的大小用树中的节点数和决策节点的复杂性度量。寻找最小树是NP-完全的(Quinlan 1986)，而我们必须使用基于启发式的局部搜索过程，在合理的时间内得到合理的树。

175

树学习算法是贪心算法，从包含全部训练数据的根开始，每一步都选择最佳划分。依赖于所选取的属性是数值属性还是离散属性，每次将数据划分成两个或n个子集。然后使用对应的子集递归地进行划分，直到不再需要划分。此时，创建一个树叶节点并标记它。

9.2.1 分类树

在用于分类的决策树，即分类树(classification tree)中，划分的优劣用不纯性度量(impurity measure)定量分析。一个划分是纯的，如果对于所有分支，划分后选择相同分支的所有

实例都属于相同的类。对于节点 m，令 N_m 为到达节点 m 的训练实例数。对于根节点，N_m 为 N。N_m 个实例中 N_m^i 个属于 C_i 类，而 $\sum_i N_m^i = N_m$。如果一个实例到达节点 m，则它属于 C_i 类的概率估计为

$$\hat{p}(C_i \mid \boldsymbol{x}, m) \equiv p_m^i = \frac{N_m^i}{N_m} \tag{9.2}$$

节点 m 是纯的，如果对于所有的 i，p_m^i 为 0 或 1。当到达节点 m 的所有实例都不属于 C_i 类时，p_m^i 为 0，而当到达节点 m 的所有实例都属于 C_i 类时，p_m^i 为 1。如果划分是纯的，则我们不需要进一步划分，并可以添加一个树叶节点，用 p_m^i 为 1 的类标记。一种度量不纯性的可能函数是熵函数(entropy)(Quinlan 1986)(参见图 9-2)。

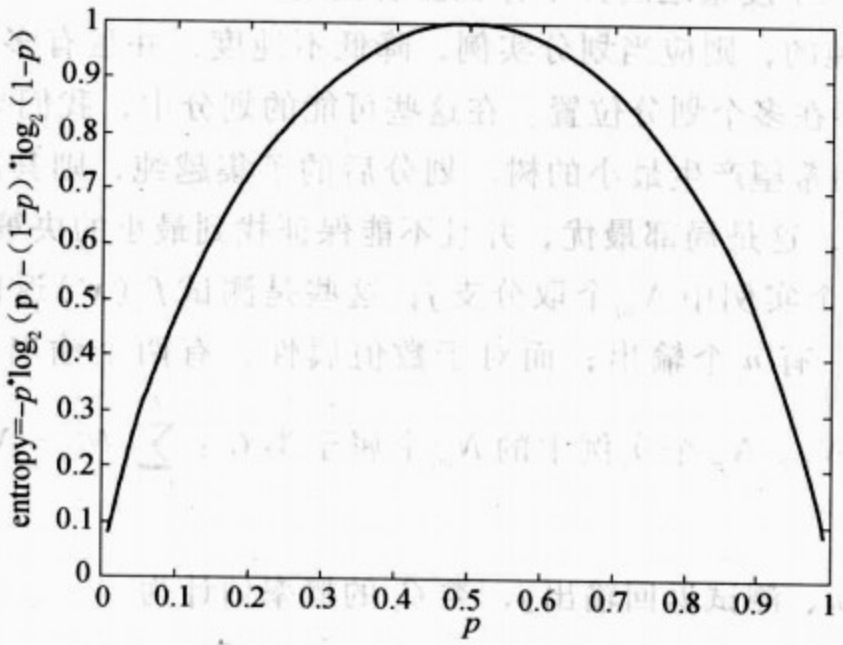


图 9-2 两类问题的熵函数

$$\mathcal{I}_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i \tag{9.3}$$

其中 $0 \log 0 \equiv 0$。在信息论中，熵是对一个实例的类代码进行编码所需要的最少位数。对于两类问题，如果 $p^1 = 1$ 而 $p^2 = 0$，则所有的实例都属于 C^1 类，并且我们什么也不需要发送，熵为 0。如果 $p^1 = p^2 = 0.5$，则我们需要发送一位通告两种情况之一，并且熵为 1。在这两个极端之间，我们可以设计编码，更可能的类用较短的编码，更不可能的类用较长的编码，每个信息使用不足一位。当存在 $K > 2$ 个类时，相同的讨论成立，并且当 $p^i = 1/K$ 时最大熵为 $\log_2 K$。

但是，熵并非是唯一可能的度量。对于两类问题，其中 $p^1 \equiv p$，$p^2 = 1 - p$，函数 $\phi(p, 1-p)$是非负函数，度量划分的不纯度，如果它满足如下性质(Devroye、Györ 和 Lugosi 1996)：

- 对于任意 $p \in [0, 1]$，$\phi(1/2, 1/2) \geq \phi(p, 1-p)$。
- $\phi(0, 1) = \phi(1, 0) = 0$。
- 当 p 在$[0, 1/2]$上时 $\phi(p, 1-p)$是递增的，而当 p 在$[1/2, 1]$上时 $\phi(p, 1-p)$是递减的。

例如

1. 熵

$$\phi(p,1-p) = -p\log_2 p - (1-p)\log_2(1-p) \tag{9.4}$$

(9.3)式是 $K>2$ 个类的推广。

2. Gini 指数(Gini index)(Breiman 等 1984)

$$\phi(p,1-p) = 2p(1-p) \tag{9.5}$$

176
~
177

3. 误分类误差

$$\phi(p,1-p) = 1 - \max(p,1-p) \tag{9.6}$$

这些都可以推广到 $K>2$ 类，并且给定损失函数，误分类误差可以推广到最小风险(习题1)。研究表明，这三个度量之间并不存在显著差别。

如果节点 m 不是纯的，则应当划分实例，降低不纯度，并且有多个属性可以用于划分。对于数值属性，可能存在多个划分位置。在这些可能的划分中，我们寻找最小化划分后的不纯度的划分，因为我们希望产生最小的树。划分后的子集越纯，则其后需要的划分(如果需要的话)就越少。当然，这是局部最优，并且不能保证找到最小的决策树。

设在节点 m，N_m 个实例中 N_{mj} 个取分支 j；这些是测试 $f_m(\boldsymbol{x}^t)$ 返回输出 j 的 $\boldsymbol{x}^t$。对于具有 n 个值的离散属性，有 n 个输出；而对于数值属性，有两个输出($n=2$)。在两种情况下，都满足 $\sum_{j=1}^{n} N_{mj} = N_m$。$N_{mj}$ 个实例中的 N_{mj}^i 个属于类 C_i：$\sum_{j=1}^{K} N_{mj}^i = N_{mj}$。类似地，$\sum_{j=1}^{n} N_{mj}^i = N_m^i$。

于是，给定节点 m、测试返回输出 j，类 C_i 的概率估计为

$$\hat{p}(C_i \mid \boldsymbol{x}, m, j) \equiv p_{mj}^i = \frac{N_{mj}^i}{N_{mj}} \tag{9.7}$$

而划分后的总不纯度为

$$\mathcal{T}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p_{mj}^i \log p_{mj}^i \tag{9.8}$$

对于数值属性，为了能够使用(9.1)式计算 p_{mj}^i，我们还需要知道该节点的 w_{m0}。在 N_m 个数据点之间，存在 $N_m - 1$ 个可能的 w_{m0}：我们不需要测试所有(无限多个)可能的点；例如，我们只需要考虑两点之间的中值就足够了。还要注意，最佳划分总是在属于不同类的两个相邻点之间。这样，我们检查每一个，并取最高纯度作为该属性的纯度。对于离散属性，不需要这种迭代。

对于所有的离散属性和数值属性，对于数值属性的所有可能划分位置，我们计算不纯度，并且选取具有最小熵的划分位置，例如在(9.8)式中度量的。于是，对于所有的不纯
178 的分支，树构造递归地、平行地继续进行，直到所有的分支都是纯的。这就是分类与回归树(classification and regression trees，CART)算法(Breiman 等，1984)、ID3 算法(Quinlan 1986)和它的扩展 C4.5(Quinlan 1993)的基本思想。算法的伪代码在图 9-3 中。

```
Generate Tree(X)
    If NodeEntropy(X)<θ_I /*(9.3)式*/
        创建一个树叶，用X中的多数类标记
        Return
    i←SplitAttribute(X)
    For x_i 的每个分支
        找出落入该分支的X_i
        Generate Tree(X_i)

SplitAttribute(X)
    MinEnt←MAX
    For 所有的属性 i=1，…，d
        If x_i 是具有 n 个值的离散属性
        按照 x_i 将X划分到X_1，…，X_n
        e←SplitEntropy(X_1，…，X_n)/*(9.8)式*/
        If e<MinEnt MinEnt←e；bestf←i
    Else /* x_i 是数值的 */
        For 所有可能的划分
            在 x_i 上将X划分成X_1，X_2
            e←SplitEntropy(X_1，X_2)
            If e<MinEnt MinEnt←e；bestf←i
Return bestf
```

图 9-3 构造分类树

也可以说，在树构造的每一步，我们选择导致不纯度降低最多的划分。不纯度的降低是到达节点 m 的数据的不纯度(9.3 式)与划分后到达其分支的数据的总熵(9.8 式)之差。

一个问题是这种划分偏向于选择具有许多值的属性。当存在许多值时，就存在许多分支，并且不纯度可能很小。例如，如果我们取训练样本的编号作为一个属性，尽管它不是一个合理的特征，但是不纯度度量将会选取它，因为这样的话，每个分支的不纯度都为 0。具有许多分支的节点是复杂的，并且背离把类判别式划分成简单决策的思想。已经提出了许多 179
方法对这样的属性加罚，并权衡不纯度下降和分支因子两个因素。

当存在噪声时，增长树直到最纯可能产生一棵非常大的、过分拟合的树。例如，假设这种情况：一个错误标记的实例混杂在一组正确标记的实例之中。为了减轻这种过分拟合，当节点变得足够纯时，树构造将终止；即，如果$\mathcal{I}<\theta_I$，则数据子集就不再划分。这意味不需要使 p_{mj}^i都恰为 0 或 1，而只需要按照某个阈值 θ_p，p_{mj}^i足够接近 0 或 1。在这种情况下，创建一个树叶节点，并将它标记为具有最大 p_{mj}^i值的类。

θ_I(或 θ_p)是复杂度参数，与非参数估计中的 h 或 k 一样。当它们较小时，方差大并且树增长较大，以正确反映训练集；而当它们较大时，方差小并且树较小，粗略地表示训练集并且可能具有较大偏倚。理想的值依赖于误分类的代价以及存储和计算开销。

一般地，建议在树叶节点存放属于每个类的后验概率，而不是用具有最大后验概率的类来标记树叶。这些概率在其后的步骤中可能是需要的。例如，在计算风险时可能需要。注意，我们不需要存放到达节点的实例或准确计数；比率就足够了。

9.2.2 回归树

回归树(regression tree)可以用几乎与分类树完全相同的方法构造，唯一的不同是适合分类的不纯性度量用适合回归的不纯性度量取代。对于节点 m，令 $\mathcal{X}_m$ 为 $\mathcal{X}$ 中到达节点 m 的子集，即它是 $\boldsymbol{x} \in \mathcal{X}$ 的满足从树根到节点 m 的所有决策节点条件的所有 $\boldsymbol{x}$。我们定义

$$b_m(\boldsymbol{x}) \begin{cases} 1 & \text{如果 } \boldsymbol{x} \in \mathcal{X}_m : \boldsymbol{x} \text{ 到达结点 } m \\ 0 & \text{否则} \end{cases} \tag{9.9}$$

180

在回归树中，划分的好坏用估计值的均方误差度量。令 g_m 为节点 m 中的估计值。

$$E_m = \frac{1}{N_m} \sum_t (r^t - g_m)^2 b_m(\boldsymbol{x}^t) \tag{9.10}$$

其中 $N_m = |\mathcal{X}_m| = \sum_t b_m(\boldsymbol{x}^t)$。

在节点中，我们使用到达该节点的实例要求的输出的均值(如果噪声太大用中值)

$$g_m = \frac{\sum_t b_m(\boldsymbol{x}^t) r^t}{\sum_t b_m(\boldsymbol{x}^t)} \tag{9.11}$$

于是，(9.10)式对应于 m 上的方差。如果在一个节点上，误差是可以接受的，即 $E_m < \theta_r$，则创建一个树叶节点，存放 g_m 值。与第8章的回归图一样，这会创建在叶边界不连续的分段常量近似。

如果误差不能接受，则到达节点 m 的数据进一步划分，使得诸分支的误差和最小。与分类一样，在每个节点上，我们寻找最小化误差的属性(和数值属性的划分阈值)，然后递归地进行上述过程。

令 $\mathcal{X}_{mj}$ 为 $\mathcal{X}_m$ 的取分支 j 的子集：$\bigcup_{j=1}^{n} \mathcal{X}_{mj} = \mathcal{X}_m$。我们定义

$$b_{mj}(\boldsymbol{x}) = \begin{cases} 1 & \text{如果 } \boldsymbol{x} \in \mathcal{X}_{mj} : \boldsymbol{x} \text{ 到达结点 } m \text{ 并取分支 } j \\ 0 & \text{否则} \end{cases} \tag{9.12}$$

g_{mj} 是到达节点 m 的分支 j 的估计值。

$$g_{mj} = \frac{\sum_t b_{mj}(\boldsymbol{x}^t) r^t}{\sum_t b_{mj}(\boldsymbol{x}^t)} \tag{9.13}$$

而划分后的误差为

$$E'_m = \frac{1}{N_m} \sum_j \sum_t (r^t - g_{mj})^2 b_{mj}(\boldsymbol{x}^t) \tag{9.14}$$

对于任意划分，误差的减少由(9.10)和(9.14)式之差给出。我们寻找这样的划分，它最大化误差的减少，或等价地，(9.14)式取最小值。将熵计算用均方误差替换，类标号用

181

平均值替换，图9-3的程序代码可以用来训练回归树。

均误方差是一种可能的误差函数；另一种是最大误差

$$E_m = \max_j \max_t |r^t - g_{mj}| b_m(\boldsymbol{x}^t) \tag{9.15}$$

使用它，我们可以保证任意实例的误差都不大于给定的阈值。

可接受的误差阈值是复杂度的函数；其值越小，产生的树越大并且过分拟合的风险越大；其值越大，拟合不足和过分光滑的可能性越大(参见图 9-4 和图 9-5)。

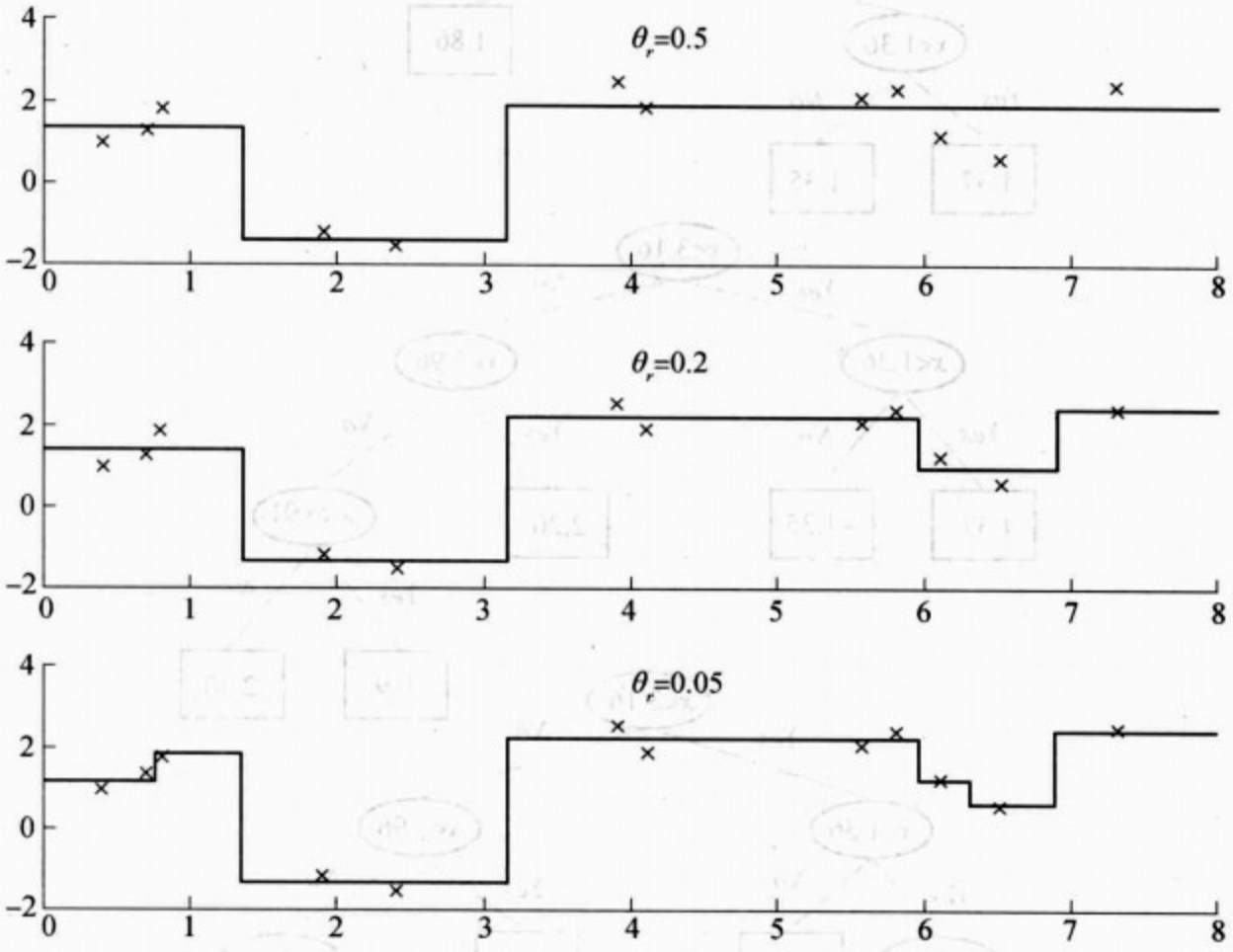


图 9-4 对于 θ_r 的不同值，回归树光滑。对应的树在图 9-5 中

类似于非参数回归中的从移动均值到移动直线，我们不是在实现常量拟合的树叶上取平均值，而是可以做线性回归拟合选定树叶上的实例：

$$g_m(\boldsymbol{x}) = \boldsymbol{w}_m^T \boldsymbol{x} + w_{m0} \tag{9.16}$$

这使得树叶上的估计依赖于 $\boldsymbol{x}$，并且产生较小的树，但是这导致树叶节点上的附加的计算开销。

9.3 剪枝

通常，如果到达一个节点的训练实例数小于训练集的某个百分比(如 5%)，则无论是否不纯或是否有错误，该节点都不进一步分裂。其基本思想是：基于过少实例的决策树导致较大方差，从而导致较大泛化误差。在树完全构造出来之前提前停止树构造称作树的先剪枝(prepruning)。

得到较小树的另一种可能做法是后剪枝(postpruning)，实践中比先剪枝效果更好。前面，我们看到树的增长很贪心，在每一步，我们做出一个决策(即产生一个决策节点)并继续进行，绝不回溯尝试其他可能的选择。唯一的例外是后剪枝，那里我们试图找出并剪掉不必要的子树。

在后剪枝中，我们让树完全增长直到所有的树叶都是纯的并具有零训练误差。然后，我们找出导致过分拟合的子树并剪掉它们。我们从最初的被标记的数据集中保留一个剪枝集(pruning set)，在训练阶段不使用。对于每棵子树，我们用一个被该子树覆盖的训练实例标记的树叶节点替换它。如果该树叶在剪枝集上的性能不比该子树差，则剪掉该子树并保留树叶节点，因为子树的附加的复杂性是不必要的；否则保留子树。

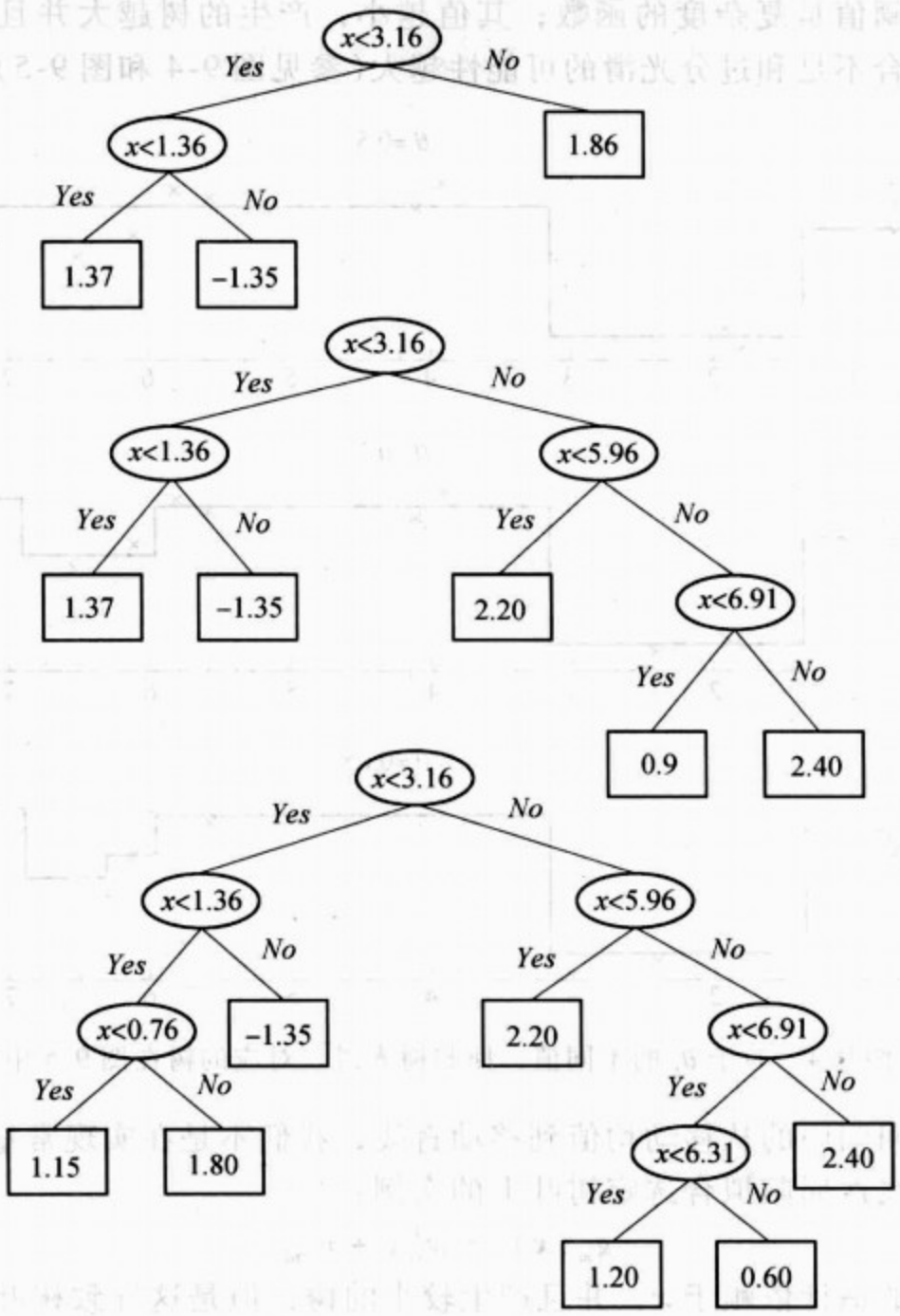


图 9-5 对于 θ_r 的不同值，实现图 9-4 的光滑的回归树

例如，在图 9-5 的第三棵树中，有一个从条件 $x<6.31$ 开始的子树。如果替换不增加剪枝集上的误差，则该子树可以用树叶节点 $y=0.9$ 替换(如第二棵树)。注意，不要把剪枝集与确认集混淆，它不同于确认集。

先剪枝与后剪枝相比，先剪枝较快，但是后剪枝通常导致更准确的树。

9.4 由决策树提取规则

决策树能够提取特征。单变量树只使用必要的变量，并且在树构建之后某些特征可能根本没有使用。我们还可以认为越靠近树根的特征从全局上讲越重要。例如，图 9-6 中的决策树使用了变量 x_1，x_2 和 x_4，但没有使用 x_3。可以使用决策树提取特征：构建一棵决策树，并取该树使用的特征作为另一种学习方法的输入。

决策树的另一优点是可解释性(interpretability)：决策树节点中的条件简单、易于理解。从树根到树叶的每条路径对应于条件的合取，这是因为为到达树叶，所有这些条件都必须满足。这些路径可以用 *IF-THEN* 规则集表示，称作规则库(rule base)。一种这样的方法是

C4.5 规则(Quinlan 1993)。

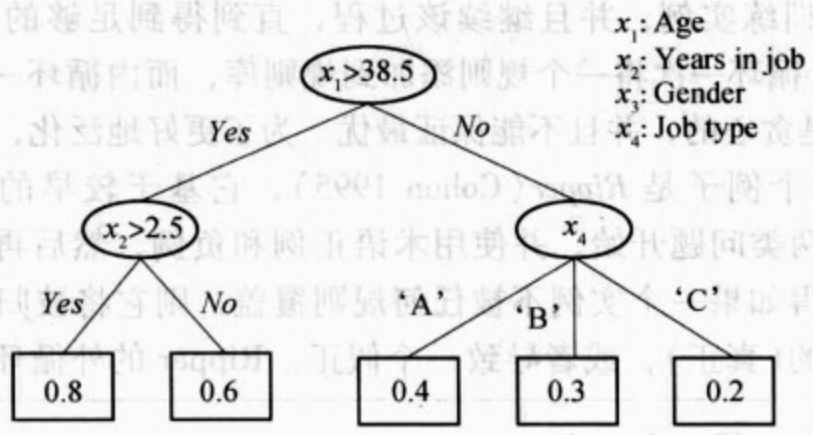


图 9-6 一棵(假想的)决策树。由根到树叶的每条路径都可以用一个合取规则表示，由该路径上决策节点定义的条件组成

例如，图 9-6 的决策树可以用如下规则集表示：

R1：IF(age > 38.5) AND(years-in-job > 2.5) THEN y = 0.8

R2：IF(age > 38.5) AND(years-in-job ≤ 2.5) THEN y = 0.6

R3：IF(age ≤ 38.5) AND(job-type = 'A') THEN y = 0.4

R4：IF(age ≤ 38.5) AND(job-type = 'B') THEN y = 0.3

R5：IF(age ≤ 38.5) AND(job-type = 'C') THEN y = 0.2

这样的规则库可以提取知识；它容易理解，并且使得领域专家可以验证从数据学习得到的模型。对于每个规则，我们可以计算被该规则覆盖的训练数据所占的百分比，即规则的支持度(rule support)。这些规则反映数据集的主要特性：它们显示了重要特征和划分位置。例如，在这个(假想的)例子中，我们看到就我们的目的(y)而言，38 岁或更年轻的人不同于 39 岁或更年长的人。并且，在后一组，工作类型区分他们；而在前一组，做一项工作的年限是最好的区分特征。

对于分类树，可能有多个树叶被标记为相同的类。在这种情况下，对应不同路径的多个合取表达式可以合并成一个析取(OR)。类区域对应于多个小区域的并，而每个小区域对应一个树叶定义的区域。例如，图 9-1 的 C_1 类可以表示为：

$$\text{IF}(x \leq w_{10})\text{OR}((x_1 > w_{10})\text{AND}(x_2 \leq w_{20}))\text{THEN } C_1$$

为了简化，可以修剪规则(pruning rule)。剪掉一棵子树对应同时从一些规则剪去一些项。可以从一个规则剪去一个项而不涉及其他规则。例如，在前面的规则集中，对于 R3，如果所有 job-type = 'A'的人无论他的年龄多大，都具有大致为 0.4 的输出，则可以对 R3 剪枝，得到

$$R3'\text{:IF(job-type = 'A') THEN } y = 0.4$$

注意，规则剪枝后可能不能再写回到树中。

9.5 由数据学习规则

正如我们刚刚看到的，产生 *IF-THEN* 规则的一种方法是训练一棵决策树，并把它转换成规则。另一种方法是直接学习规则。规则归纳(rule induction)类似于树归纳，唯一的区别在于规则归纳进行深度优先搜索，并且一次产生一条路径(规则)；而树归纳进行宽度优先搜索，并且同时产生所有路径。

一次学习一个规则。每个规则是离散或数值属性上条件的合取(与决策树一样)，并且这些条件一次添加一个，以优化某个标准，如最小化熵。我们说规则覆盖(cover)一个实例，

如果该实例满足规则的所有条件。一旦规则形成并被剪枝，就将它添加到规则库中，从训练集中删除被它覆盖的所有训练实例，并且继续该过程，直到得到足够的规则。这称作顺序覆盖(sequential covering)。外循环一次将一个规则添加到规则库，而内循环一次将一个条件添加到当前规则中。这些步骤都是贪心的，并且不能保证最优。为了更好地泛化，两个循环都有剪枝步。

规则归纳算法的一个例子是 *Ripper*(Cohen 1995)，它基于较早的算法 *Irep*(Fürnkrantz 和 Widmer 1994)。我们从两类问题开始，并使用术语正例和负例，然后再推广到 $K>2$ 类。添加规则旨在解释正例，使得如果一个实例不被任何规则覆盖，则它将被归到负类。这样，当规则匹配时，它或者是正确的(真正)，或者导致一个假正。Ripper 的外循环的伪代码在图 9-7 中。

```
Ripper(Pos, Neg, k)
    RuleSet←LearnRuleSet(Pos, Neg)
    For k times
        RuleSet←OptimizeRuleSet(RuleSet, Pos, Neg)
LearnRuleSet(Pos, Neg)
    RuleSet←∅
    DL←DescLen(RuleSet, Pos, Neg)
    Repeat
        Rule←LearnRule(Pos, Neg)
        将 Rule 添加到 RuleSet
        DL' ←DescLen(RuleSet, Pos, Neg)
        If DL' > DL + 64
            PruneRuleSet(RuleSet, Pos, Neg)
            Return RuleSet
        If DL' < DL DL←DL'
            从 Pos 和 Neg 删除被 Rule 覆盖的实例
    Until Pos = ∅
    Return RuleSet
PruneRuleSet(RuleSet, Pos, Neg)
    For 每个 Rule ∈ RuleSet，按相反次序
        DL←DescLen(RuleSet, Pos, Neg)
        DL' ←DescLen(RuleSet-Rule, Pos, Neg)
        If DL' < DL 从 RuleSet 中删除 Rule
    Return RuleSet
OptimizeRuleSet(RuleSet, Pos, Neg)
    For 每个 Rule ∈ RuleSet
        DL0←DescLen(RuleSet, Pos, Neg)
        DL1←DescLen(RuleSet-Rule +
                ReplaceRule(RuleSet, Pos, Neg), Pos, Neg)
        DL2←DescLen(RuleSet-Rule +
                ReviseRule(RuleSet, Rule, Pos, Neg), Pos, Neg)
        If DL1 = min(DL0, DL1, DL2)
            从 RuleSet 中删除 Rule 并且
            添加 ReplaceRule(RuleSet, Pos, Neg)
        Else If DL2 = min(DL0, DL1, DL2)
            从 RuleSet 中删除 Rule 并且
            添加 ReviseRule(RuleSet, Rule, Pos, Neg)
Return RuleSet
```

图 9-7 学习规则的 Ripper 算法。只给出了外循环，内循环与在决策树中添加一个节点类似

在 Ripper 中，条件被添加到规则中，以最大化 Quinlan 的 Foil 算法(1990)使用的信息增益度量。假设我们有规则 R，并且 R'是添加一个条件后的候选规则。增益的改变定义为

$$Gain(R',R) = s \cdot \left(\log_2 \frac{N'_+}{N'} - \log_2 \frac{N_+}{N} \right) \tag{9.17}$$

其中 N 是被 R 覆盖的实例数，而 N_+是其中的真正例数。类似地，N'是被 R'覆盖的实例数，N'_+是其中的真正例数。s 是 R 中的真正例并且增加条件之后在 R'也是真正实例的实例数。根据信息理论，增益的变化度量对一个正例编码所需位的减少。

向规则增加条件直到它不再覆盖负例。一旦规则形成，就以相反的次序对它剪枝，以便找到最大化规则价值度量(rule value metric)的规则

$$rvm(R) = \frac{p - n}{p + n} \tag{9.18}$$

其中 p 和 n 分别是剪枝集中的真正例和假正例数。剪枝集是数据的三分之一，已经使用三分之二的数据作为增长集。

一旦规则形成并被剪枝，就从训练集中删除被规则覆盖的所有正的和负的训练实例。如果还有正的实例，则继续进行规则归纳。在存在噪声的情况下，即当规则不能解释足够多的实例时，我们可能提前中止归纳。为了度量规则的价值，使用最小描述长度(参见 4.8 节)(Quinlan 1995)。典型地，如果规则的描述长度不短于它所解释的实例的描述长度，则我们停止。规则库的描述长度是规则库中所有规则的描述长度之和，加上不被规则库覆盖的实例的描述长度。当规则的描述长度比迄今得到的最佳描述长度多 64 位时，Ripper 停止添加规则。一旦学到了规则库，我们就以逆序忽略规则，看是否能够删除它们而不增加描述长度。

规则库中的规则在学习之后也要优化。对一个规则，Ripper 考虑两种可供选择的方案：一种是置换规则，从空规则开始，增长然后剪枝。第二种是修订规则，从规则开始，增长然后剪枝。这两个规则与原规则比较，并将三个中的最短者添加到规则库中。规则库的这种优化进行 k 次，通常进行两次。

当存在 $K>2$ 个类时，将这些类按照它们的先验概率排序，使得 C_1 的先验概率最低，C_K 的先验概率最高。然后定义一系列两类问题。开始，属于 C_1 的实例为正例，其他类的实例都是负例。学习 C_1 的规则后，删除它的所有实例，学习将 C_2 与 C_3，…，C_K 分离开来。重复该过程，直到只剩下 C_K。空的缺省规则标记为 C_K，使得如果一个实例不被任何规则覆盖，则将它指派到 C_K。

对于大小为 N 的训练集，Ripper 的复杂度为$O(N \log^2 N)$，并且可以用于很大的训练集(Dietterich 1997)。学习的规则是命题规则(propositional rule)。更准确地说，是条件中包含变量的一阶规则(first-order rule)，称作谓词(predicate)。谓词是一个函数，依赖于其变元的值，它返回真或假。因此，谓词可以定义属性值之间的关系，而命题不能(Mitchell 1997)：

187
≀
189

IF Father(y,x) AND Female(y) THEN Daughter(x,y)

在逻辑程序设计语言(如 Prolog)中，这种规则可以看作程序，而从数据中学习它们称作归纳逻辑程序设计(inductive logic programming)。一种这样的算法是 Foil(Quinlan 1990)。

将一个值指派到一个变量称作绑定(binding)。如果训练集中存在到变量的绑定集，则称规则匹配。学习一阶规则类似于学习命题规则，外循环添加规则，而内循环相规则添加条件，在内循环结束时进行规则剪枝。不同的是，在内循环中，每一步我们考虑增加一个谓词

(而不是命题)，并检查规则的性能提高(Mitchell 1997)。为了评估规则的性能，我们考虑变量的所有可能绑定，对训练集中正的和负的绑定计数，并使用，例如(9.17)式。在学习一阶规则时，我们使用谓词而不是命题，因此这些谓词应当事先定义，并且训练集是已知为真的谓词集。

9.6 多变量树

在构造单变量树时，划分时只使用一个输入维。在构造多变量树(multivariate tree)时，在每个决策节点都可以使用所有的输入维，因此更加一般。当所有的输入都是数值属性时，二元线性多变量节点定义为

$$f_m(\boldsymbol{x}):\boldsymbol{w}_m^T\boldsymbol{x}+w_{m0}>0 \tag{9.19}$$

因为线性多变量节点取加权和，因此离散属性应当用0/1哑数值变量表示。(9.19)式定义了一个具有任意方向的超平面(参见图9-8)。从根到树叶的路径上的相继节点进一步划分实例，而叶节点定义输入空间上的多面体。具有数值特征的一元节点是一种特例，所有的w_{mj}除一个之外均为0。这样，(9.1)式的单变量数值节点也定义了一个线性判别式，但是与轴x_j正交于w_{m0}，与其他轴x_i平行。因此，我们看到在单变量节点有d个可能的方向($\boldsymbol{w}_m$)和N_m-1个可能的阈值($-w_{m0}$)，使得穷举搜索是可能的。在多变量节点，有$2^dC_{N_m}^d$个可能
190 的超平面(Murthy、Kasif和Salzberg 1994)，并且不再可能进行穷举搜索。

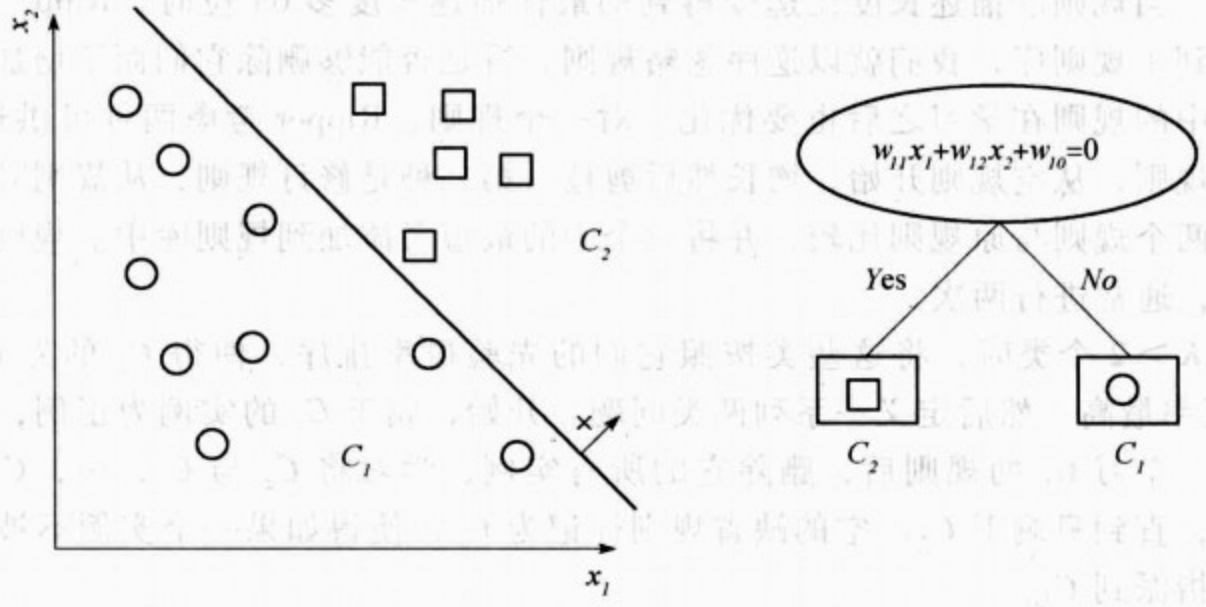


图9-8 线性多变量决策树。线性多变量节点可以安放任意超平面，因而更一般，而单变量节点限于平行于轴的划分

当我们从单变量节点过渡到线性多变量节点时，节点变得更灵活。使用非线性多变量节点，还可以更加灵活。例如，使用二次方程，我们有

$$f_m(\boldsymbol{x}):\boldsymbol{x}^T\mathbf{W}_m\boldsymbol{x}+\boldsymbol{w}_m^T\boldsymbol{x}+w_{m0}>0 \tag{9.20}$$

Guo和Gelfand(1992)提出使用多层感知器(第11章)。多层感知器是非线性基函数的线性和，是另一种产生非线性决策节点的方法。还一种可能性是使用球形节点(sphere node)(Devroye、Györfi和Lugosi 1996)

$$f_m(\boldsymbol{x}):\|\boldsymbol{x}-\boldsymbol{c}_m\|\leq\alpha_m \tag{9.21}$$

其中 $\boldsymbol{c}_m$ 是球心，α_m 是半径。

已经提出了一些学习用于分类的多变量决策树的算法：最早的算法是 CART 算法的多变量版本(Breiman 等 1984)，它逐一对权重 w_{mj} 进行微调来降低不纯度。CART 还包含一个预处理步骤，通过子集选择降低维度(第 6 章)并降低节点的复杂度。一种对 CART 扩展的算法 191
是 OC1 算法(Murthy、Kasif 和 Salzberg 1994)。一种可能的方法(Loh 和 Vanichsetakul 1988)是假设所有的类是高斯的，具有共同的协方差矩阵，因此具有分离每个类的线性判别式(第 5 章)。在这种情况下，具有 K 个类，每个节点具有 K 个分支，而每个分支携带一个将每个类与其他类分开的线性判别式。Brodley 和 Utgofi(1995)提出了一种方法，这种方法训练线性判别式以最小化分类误差(第 10 章)。Guo 和 Gelfand(1992)提出了一种方法，将 $K>2$ 个类划分为两个超群，然后学习二元多变量树。Loh 和 Shih(1997)使用 2-均值聚类(第 7 章)将数据聚成两组。一旦类聚成两组，Yıldız 和 Alpaydın(2000)就使用 LDA(第 6 章)找出判别式。

任何分类器都近似一个从假设类中选取一个假设的实际(未知的)判别式。当我们使用单变量节点时，我们的近似使用分段的、平行于轴的超平面。使用线性多变量节点，我们可以使用任意的超平面，并且使用较少的节点得到更好的近似。如果潜在的判别式是曲的，非线性节点更好。分支因子确定节点定义的判别式的个数，具有类似效果。具有两个分支的二元决策节点定义一个将输入空间一分为二的判别式。n-路节点将输入空间划分成 n 部分。这样，节点、分支因子和树大小之间存在相关性。使用简单节点和较低的分支因子可以得到一棵大树。但是，这样的树(如具有单变量二元节点)可解释性更好。线性多变量节点更难解释。更复杂的节点也需要更多的数据，并且随着我们沿树向下，数据越来越少，更容易过分拟合。如果节点复杂并且树比较小，那么我们也就失去了通过树想要得到的主要东西——将问题划分成一系列简单问题。毕竟，我们可以在根节点具有一个非常复杂的分类器，它区分所有的类，但这样的话，这就不是一棵树了！

9.7 注释

自从凯撒将一个复杂的问题(如高卢人问题)分解成一组较简单的问题以来，分治一直 192
被作为一种启发式方法频繁使用。在计算机科学中，频繁地使用树将复杂度从线性降低到对数时间。Breiman 等 1984 使得决策树在统计学流行，Quinlan 1986、1993 使得决策树在在机器学习中很流行。多变量树归纳方法最近才开始流行，Yıldız 和 Alpaydın 2000 给出了综述和许多数据集上的比较。许多研究者(如 Guo 和 Gelfand 1992)将树的简单性与多层感知器的准确性结合在一起(第 11 章)。然而，许多研究表明单变量树相当准确、具有很好的可解释性，并且线性(非线性)多变量节点带来的附加的复杂度很难被认为是合理的。

杂变量决策树(omnivariate decision tree)(Yıldız 和 Alpaydın 2001)是一种混合树结构，其中树可以具有单变量、线性多变量和非线性多变量节点。其基本思想是在树构造期间，每个决策节点对应于到达该节点的训练数据子集定义的一个不同的子问题，不同的模型可能更合适，应当找出和使用合适的模型。到处都用相同类型的节点相当于假定输入空间的所有部分都具有相同的归纳偏倚。在杂变量树中，在每个节点，训练并在确认集上使用统计检验(第 14 章)比较不同类型的候选节点，确定哪一个泛化性能最好。除非复杂的决策节点表现出显

著较高的准确率，否则就选取较简单的候选节点。结果表明，在树构造的早期更靠近树根的地方使用较复杂的节点，并且随着我们沿树向下，简单的单变量节点就足够了。随着我们越来越靠近树叶，问题越来越简单，同时，数据越来越少。在这种情况下，复杂的节点过分拟合，并且被统计检验拒绝。随着我们沿树向下，节点的个数指数增加。因此，大部分节点是单变量的，并且总体复杂度增加不太多。

决策树更多地用于分类而不是回归。它们非常流行：它们的学习和响应速度都很快，并且在许多领域都很准确(Murthy 1998)。由于它们的可解释性，甚至在有更准确的方法时，决策树仍是首选。当决策树写成 IF-THEN 规则集时，树可以理解，并且可以被具有应用领域知识的专家验证。

193 通常，在使用更复杂算法之前，建议先试验决策树，并将它的准确率作为性能基准。树分析还能帮助我们理解重要特征，单变量树还可以用于自动特征提取。单变量树的另一个重要优点是它可以使用数值和离散特征，而不需要将一种类型转换成另一种。

决策树是非参数方法，类似于第 8 章讨论的方法，但是存在一些不同之处：

- 每个树叶对应于一个“箱”，只不过箱不必具有相同的大小(如 Parzen 窗口)或相同个数的实例(如 k-最近邻)。
- 箱的划分不仅仅根据输入空间中的相似度，而是需要通过熵或均方误差，使用输出信息。
- 决策树的另一个优点是仅通过少量比较就能找到树叶(箱)。
- 决策树一旦构造就不需要存放所有的训练集，而只需要存放树的结构、决策节点的参数和树叶节点的输出值。与需要存储所有训练实例的基于核的或基于近邻的方法相比，这意味决策树的空间复杂度也非常小。

使用决策树，一个类不必具有所有实例都匹配的单个描述。一个类可以具有多个可能的描述，它们甚至可能在输入空间中不相交。

树不同于上一章讨论的统计模型。树直接地对分离类实例的判别式编码，而不必携带许多关于这些实例在该区域中如何分布的信息。决策树是基于判别式的 (discriminant-based)，而统计学方法是基于似然的 (likelihood-based)，它们在使用贝叶斯规则和计算判别式之前显式地估计 $p(\boldsymbol{x} \mid C_i)$。基于判别式的方法绕过类密度估计，直接估计判别式。在其后几章，

194 我们将进一步讨论基于判别式的方法。

9.8 习题

1. 将 Gini 指数(9.5)式和误分类误差(9.6)式推广到 $K>2$ 个类。考虑损失函数，将误分类误差推广到风险。
2. 对于数值属性，我们可以不用二元划分，而通过两个阈值和三个分支使用三元划分

$$x_j < w_{ma}, w_{ma} \leqslant x_j < w_{mb}, x_j \geqslant w_{mb}$$

修改决策树归纳方法，学习两个阈值 w_{ma} 和 w_{mb}。与二元节点相比，这种节点的优缺点是什么？

3. 提出一种带回溯的树归纳算法。
4. 在产生单变量树时，具有 n 个可能值的离散属性可以用 n 个 0/1 哑变量表示，并将它们

看作是 n 个分别的数值属性。这种方法的优缺点是什么？

5. 为球形树(9.21)式推导一个学习算法。将它推广到椭球形树。
6. 在回归树中，我们提到在树叶节点不是计算均值，而是可以做线性回归拟合，并使树叶上的响应依赖于输入。对分类树提出一种类似的方法。
7. 为回归提出一种规则归纳算法。

9.9 参考文献

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group.

Brodley, C. E., and P. E. Utgoff. 1995. "Multivariate Decision Trees." *Machine Learning* 19: 45 – 77.

Cohen, W. 1995. "Fast Effective Rule Induction." In *Twelfth International Conference on Machine Learning*, ed. A. Prieditis and S. J. Russell, 115 – 123. San Mateo, CA: Morgan Kaufmann.

Devroye, L., L. Györfi, and G. Lugosi. 1996. *A Probabilistic Theory of Pattern Recognition*. New York: Springer.

Dietterich, T. G. 1997. "Machine Learning Research: Four Current Directions." *AI Magazine* 18: 97 – 136. 195

Fürnkrantz, J., and G. Widmer. 1994. "Incremental Reduced Error Pruning." In *Eleventh International Conference on Machine Learning*, ed. W. Cohen and H. Hirsh, 70 – 77. San Mateo, CA: Morgan Kaufmann.

Guo, H., and S. B. Gelfand. 1992. "Classification Trees with Neural Network Feature Extraction." *IEEE Transactions on Neural Networks* 3: 923 – 933.

Loh, W. - Y., and Y. S. Shih. 1997. "Split Selection Methods for Classification Trees." *Statistica Sinica* 7: 815 – 840.

Loh, W. - Y., and N. Vanichsetakul. 1988. "Tree- Structured Classification via Generalized Discriminant Analysis." *Journal of the American Statistical Association* 83: 715 – 725.

Mitchell, T. 1997. *Machine Learning*. New York: McGraw-Hill.

Murthy, S. K. 1998. "Automatic Construction of Decision Trees from Data: A Multi- Disciplinary Survey." *Data Mining and Knowledge Discovery* 4: 345 – 389.

Murthy, S. K., S. Kasif, and S. Salzberg. 1994. "A System for Induction of Oblique Decision Trees." *Journal of Artificial Intelligence Research* 2: 1 – 32.

Quinlan, J. R. 1986. "Induction of Decision Trees." *Machine Learning* 1: 81 – 106.

Quinlan, J. R. 1990. "Learning Logical Definitions from Relations." *Machine Learning* 5: 239 – 266.

Quinlan, J. R. 1993. *C4. 5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. 1995. "MDL and Categorical Theories (continued)." In *Twelfth International Conference on Machine Learning*, ed. A. Prieditis and S. J. Russell, 467 – 470. San Mateo, CA: Morgan Kaufmann.

Yıldız, O. T., and E. Alpaydin. 2000. "Linear Discriminant Trees." In *Seventeenth International Conference on Machine Learning*, ed. P. Langley, 1175 – 1182. San Francisco: Morgan Kaufmann.

Yıldız, O. T., and E. Alpaydin. 2001. "Omnivariate Decision Trees." *IEEE Transactions on Neural Networks* 12: 1539 – 1546. 196

第 10 章　线性判别式

在线性判别式中，我们假定类的实例是线性可分的。这是一种基于判别式的方法，它直接估计判别式的参数，而不必先估计各种概率。本章，我们会看到具有不同偏倚的不同学习算法，从给定的有标记的训练样本学习这种线性判别式。

10.1　引言

在前面的章节中，对于分类，我们定义了一组判别式函数 $g_j(\boldsymbol{x})$，$j=1$，…，K，并且如果 $g_i(\boldsymbol{x}) = \max_{j=1}^{K} g_j(\boldsymbol{x})$，我们就选择 C_i。

前面，在我们讨论分类方法时，我们首先估计先验概率 $\hat{p}(C_i)$ 和类似然 $\hat{p}(\boldsymbol{x} \mid C_i)$，再使用贝叶斯规则计算后验密度。然后，我们使用后验密度定义判别式函数，例如

$$g_i(\boldsymbol{x}) = \log \hat{p}(C_i \mid \boldsymbol{x})$$

这称作基于似然的分类（likelihood-based classification），并且我们在前面已经讨论了估计类似然 $P(\boldsymbol{x} \mid C_i)$ 的参数（第 5 章）、半参数（第 7 章）和非参数（第 8 章）方法。

现在，我们讨论基于判别式的分类（discriminant-based classification），这里我们绕过似然或后验概率的估计，直接为判别式假定模型。基于判别式的方法对类之间的判别式形式进行假设，而不对密度（例如，是否是高斯分布）、输入是否相关等知识做任何假设。基于判别式的方法被称为非参数（nonparametric）方法，这里参数意指类似然密度的参数。

我们为判别式定义一个模型

$$g_i(\boldsymbol{x} \mid \Phi_i)$$

显式地用参数 Φ_i 的集合参数化。这与基于似然的模式不同。基于似然的方法在定义似然密度时具有隐式参数。这是不同的归纳偏倚：我们对判别式的形式进行假设，而不是对密度的形式进行假设。

学习是优化模型参数 Φ_i，最大化给定类标号的训练集上的分类准确率。这不同于基于似然的方法。基于似然的方法分别为每个类搜索最大化样本似然的参数。

在基于判别式的方法中，我们并不关注正确地估计类区域中的密度；我们所关注的是正确估计类区域之间的边界（boundary）。基于判别式方法的创导者（如 Cherkassky 和 Mulier 1998）指出，估计类密度比估计类判别式更困难，并且为解决较容易的问题而解决困难的问题并无意义。当然，仅当判别式可以用简单函数近似时才确实如此。

本章，我们关注最简单的情况，其中判别式是 $\boldsymbol{x}$ 的线性函数：

$$g_i(\boldsymbol{x} \mid \boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0} \tag{10.1}$$

线性判别式（linear discriminant）经常使用，主要是由于它的简单性，即它的空间和时间复杂度都是 $\mathcal{O}(d)$。线性模型容易理解：最终的输出是若干因素的加权和。权重的大小显示

了这些因素的重要性，而它们的符号显示其作用的正负。大部分函数是可加的，因为输出是若干属性作用的加权和，其中权重可能是正的(加强)或负的(抑制)。例如，当一位顾客申 198
请信用卡时，金融机构计算申请者的信用得分。得分一般是多个属性作用之和。例如，年薪的作用为正(较高的年薪增加得分)。

在许多应用中，线性判别式相当准确。例如，我们知道当类是高斯的，具有相同的协方差矩阵时，最佳的判别式是线性的。然而，即使该假设不成立，也可以使用线性判别式，并且不必对类密度做任何假设就能计算模型参数。在试用更复杂的模型，确保附加的复杂性是合理的之前，我们将一直使用线性判别式。

正如我们一直做的那样，我们把寻找线性判别式函数问题归结为搜索最小化某个误差函数的参数值问题。我们尤其关注优化准则函数的梯度(gradient)方法。

10.2 推广线性模型

当线性判别式不够灵活时，我们可以提高复杂度，使用二次判别式(quadratic discriminant)函数

$$g_i(\boldsymbol{x} \mid \mathbf{W}_i, \boldsymbol{w}_i, w_{i0}) = \boldsymbol{x}^T \mathbf{W}_i \boldsymbol{x} + \boldsymbol{w}_i \boldsymbol{x} + w_{i0} \tag{10.2}$$

但是，这种方法的复杂度是$\mathcal{O}(d^2)$，并且我们还会遇到偏倚和方差的两难选择：尽管二次模型更一般，但是它需要更大的训练集，并且在小样本上可能过分拟合。

一种等价的方法是通过增加高阶项(higher-order term)，又称乘积项(product term)，对输入进行预处理。例如，对于两个输入x_1和x_2，我们可以定义新变量

$$z_1 = x_1, z_2 = x_2, z_3 = x_1^2, z_4 = x_2^2, z_5 = x_1 x_2$$

并取$\boldsymbol{z} = [z_1, z_2, z_3, z_4, z_5]^T$为输入。定义在五维$\boldsymbol{z}$空间上的线性函数对应二维$\boldsymbol{x}$空间上的非线性函数。替代在原空间定义非线性函数(判别式或回归)，我们需要做的是定义到新空间的、合适的非线性变换，其中新空间上的函数可以是线性的。 199

判别式可以表示成

$$g_i(\boldsymbol{x}) = \sum_{j=1}^{k} w_j \phi_{ij}(\boldsymbol{x}) \tag{10.3}$$

其中$\phi_{ij}(\boldsymbol{x})$是基函数(basis function)。例子如下

- $\sin(x_1)$
- $\exp(-(x_1 - m)^2/c)$
- $\exp(-\|\boldsymbol{x} - \boldsymbol{m}\|^2/c)$
- $\log(x_2)$
- $1(x_1 > c)$
- $1(ax_1 + bx_2 > c)$

其中m，a，b，c是标量，$\boldsymbol{m}$是d维向量，而当b为真时$1(b)$返回1，否则返回0。将非线性函数表示成非线性基函数的线性和的想法并非新想法，并且最初称作潜函数(potential function)(Aizerman、Braverman和Rozonoer 1964)。在10.9节，我们讨论使用这种基函数的支持向量机。多层感知器(第11章)和径向基函数(第12章)具有进一步的优点，可以在学习时调整基函数的参数。

10.3 线性判别式的几何意义

10.3.1 两类问题

让我们从最简单的两类问题开始。在这种情况下，一个判别式函数就足够了：

$$\begin{aligned} g(\boldsymbol{x}) &= g_1(\boldsymbol{x}) - g_2(\boldsymbol{x}) \\ &= (\boldsymbol{w}_1^T\boldsymbol{x} + w_{10}) - (\boldsymbol{w}_2^T\boldsymbol{x} + w_{20}) \\ &= (\boldsymbol{w}_1 - \boldsymbol{w}_2)^T\boldsymbol{x} + (w_{10} - w_{20}) \\ &= \boldsymbol{w}^T\boldsymbol{x} + w_0 \end{aligned}$$

200 并且如果 $g(x)>0$，我们选择 C_1，否则选择 C_2。

这定义了一个超平面，其中 $\boldsymbol{w}$ 是权重向量(weight vector)，w_0 是阈值(threshold)。后者称作阈值是因为规则可以改写为：如果 $\boldsymbol{w}^T\boldsymbol{x}>-w_0$，选择 C_1，否则选择 C_2。超平面将输入空间划分成两个半空间：C_1 的决策区域 R_1 和 C_2 的决策区域 R_2。R_1 中的任何 $\boldsymbol{x}$ 都在超平面的正(positive)侧，而 R_2 中的任何 $\boldsymbol{x}$ 都在超平面的负(negative)侧。当 $\boldsymbol{x}$ 为 0 时，$g(\boldsymbol{x}) = w_0$ 并且如果我们有 $w_0>0$，则原点在超平面的正侧，如果 $w_0<0$，则原点在超平面的负侧，而如果 $w_0=0$，则超平面经过原点(见图 10-1)。

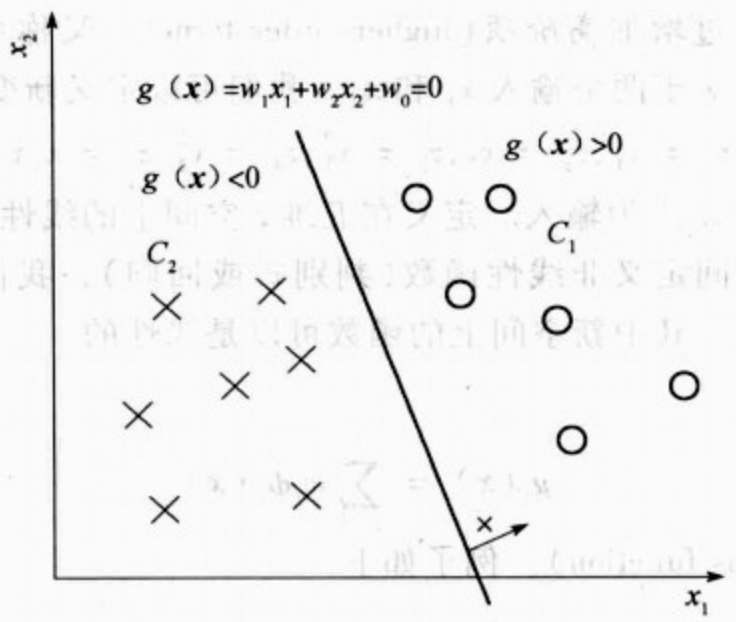


图 10-1 在二维情况下，线性判别式是一条将两个类的实例分开的直线

取决策面上的两个点 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$(即 $g(\boldsymbol{x}_1)=g(\boldsymbol{x}_2)=0$)，则

$$\boldsymbol{w}^T\boldsymbol{x}_1 + w_0 = \boldsymbol{w}^T\boldsymbol{x}_2 + w_0$$

$$\boldsymbol{w}^T(\boldsymbol{x}_1 - \boldsymbol{x}_2) = 0$$

并且我们看到 $\boldsymbol{w}$ 是超平面上的任意向量的法线。让我们将 $\boldsymbol{x}$ 改写为(Duda、Hart 和 Stork 2001)

201

$$\boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$

其中 $\boldsymbol{x}_p$ 是 $\boldsymbol{x}$ 到超平面的法向投影，而 r 给出 $\boldsymbol{x}$ 到超平面的距离，如果 $\boldsymbol{x}$ 在负侧，则它为负；如果 $\boldsymbol{x}$ 在正侧，则它为正(参见图 10-2)。计算 $g(\boldsymbol{x})$ 并注意 $g(\boldsymbol{x}_p)=0$，我们有

$$r = \frac{g(\boldsymbol{x})}{\|\boldsymbol{w}\|} \tag{10.4}$$

于是，我们看到超平面到原点的距离为

$$r_0 = \frac{w_0}{\|\boldsymbol{w}\|} \tag{10.5}$$

这样，w_0 决定超平面关于原点的位置，而 $\boldsymbol{w}$ 决定它的方向。

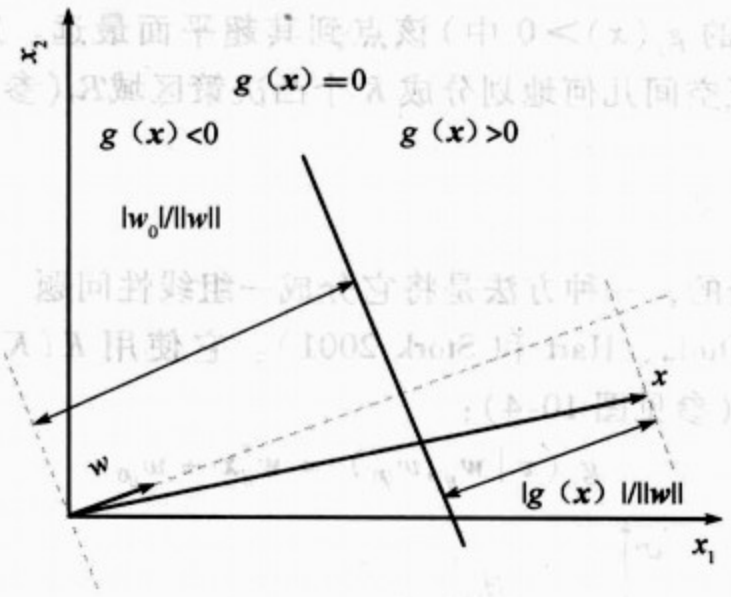


图 10-2 线性判别式的几何解释

10.3.2 多类问题

当存在 $K>2$ 个类时，有 K 个判别式函数。当它们都是线性的时，我们有

$$g_i(\boldsymbol{x} \mid \boldsymbol{w}_i, w_{i0}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} \tag{10.6}$$

202

稍后，我们将讨论学习，但是现在我们假定参数 $\boldsymbol{w}_i$ 和 w_{i0} 的计算使得对于训练集中的所有 $\boldsymbol{x}$，

$$g_i(\boldsymbol{x} \mid \boldsymbol{w}_i, w_{i0}) = \begin{cases} >0 & \text{如果 } \boldsymbol{x} \in C_i \\ \leqslant 0 & \text{否则} \end{cases} \tag{10.7}$$

使用这种判别函数相当于假设所有的类都是线性可分的(linearly separable)；即对于每个类 C_i，存在一个超平面 H_i，使得所有的 $\boldsymbol{x} \in C_i$ 都在它的正侧，所有的 $\boldsymbol{x} \in C_j$，$j \neq i$ 都在它的负侧(参见图 10-3)。

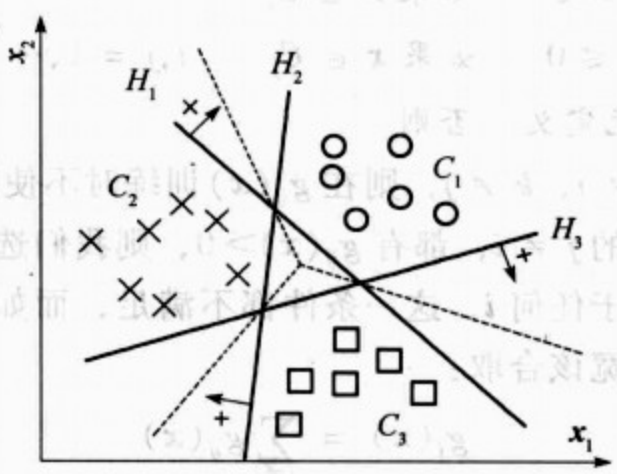


图 10-3 在线性分类，每个超平面 H_i 将 C_i 类的实例与其他类的实例分开。为了做到这一点，类应当是线性可分的。虚线是线性分类器的归约后的边界

在检验时，给定 $\boldsymbol{x}$，理想情况下应当只有一个 $g_j(\boldsymbol{x})(j=1, \cdots, K)$ 大于 0，而其他的都小于 0。但是，并非总是如此：这些超平面的正的半个空间可能重叠，或者说可能存在所有的 $g_j(\boldsymbol{x})$ 都小于 0 的案例。这些案例可以看作拒绝（reject）案例，但是通常的方法是将 $\boldsymbol{x}$ 指派到具有最大判别式值的类：

$$\text{选择 } C_i \text{ 如果 } g_i(\boldsymbol{x}) = \max_{j=1}^{K} g_j(\boldsymbol{x}) \tag{10.8}$$

203 注意，$|g_i(\boldsymbol{x})| / \|\boldsymbol{w}_i\|$ 是从输入点 $\boldsymbol{x}$ 到超平面的距离。假定 $\boldsymbol{w}_i$ 具有类似的长度，这将该点指派到这个类，（在所有的 $g_j(\boldsymbol{x})>0$ 中）该点到其超平面最远。这称作线性分类器（linear classifier），并且它将特征空间几何地划分成 K 个凸决策区域 $\mathcal{R}_i$（参见图 10-3）。

10.4 逐对分离

如果类不是线性可分的，一种方法是将它分成一组线性问题。一种可能的方法是逐对分离（pairwise separation）（Duda、Hart 和 Stork 2001）。它使用 $K(K-1)/2$ 个线性判别式 $g_{ij}(\boldsymbol{x})$，每对不同的类一个（参见图 10-4）：

$$g_{ij}(\boldsymbol{x} \mid \boldsymbol{w}_{ij}, w_{ij0}) = \boldsymbol{w}_{ij}^T \boldsymbol{x} + w_{ij0}$$

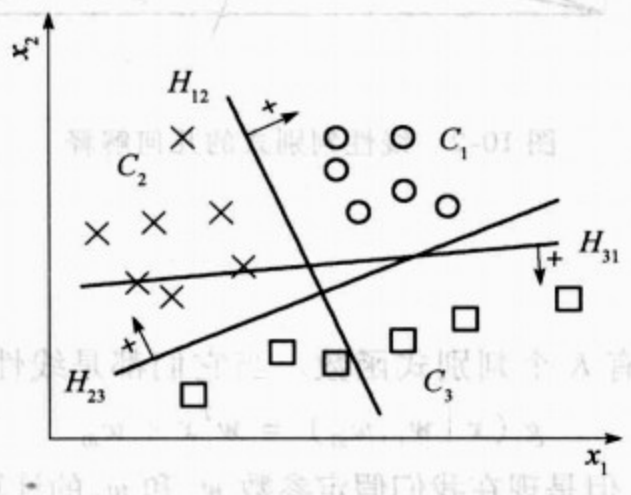


图 10-4 在逐对线性分离中，每一对类有一个分离超平面。一个输入被指派到 C_1，它应当在 H_{12} 和 H_{13} 的正侧（H_{13} 的正侧是 H_{31} 的负侧）；我们不考虑 H_{23} 的值。在这种情况下，C_1 不是关于其他类线性可分的，但却是逐对线性可分的

参数 $w_{ij}(j \neq i)$ 在训练时计算，使得

$$g_{ij}(\boldsymbol{x}) = \begin{cases} >0 & \text{如果 } \boldsymbol{x} \in C_i \\ \leqslant 0 & \text{如果 } \boldsymbol{x} \in C_j \\ \text{无定义} & \text{否则} \end{cases} \quad i,j = 1, \cdots, K \text{ 并且 } i \neq j \tag{10.9}$$

204 也就是说，如果 $\boldsymbol{x}^t \in C_k$，$k \neq i$，$k \neq j$，则在 $g_{ij}(\boldsymbol{x})$ 训练时不使用 $\boldsymbol{x}^t$。

在检验时，如果对任意的 $j \neq i$，都有 $g_{ij}(\boldsymbol{x})>0$，则我们选择 C_i。

在许多情况下，可能对于任何 i，这一条件都不满足，而如果我们不想丢弃这种情况的话，我们可以用如下和式放宽该合取：

$$g_i(\boldsymbol{x}) = \sum_{j \neq i} g_{ij}(\boldsymbol{x}) \tag{10.10}$$

即使这些类不是线性可分的，如果这些类是逐对线性可分的（这种情况可能性更大），则可以使用逐对分离，导致类的非线性分离（参见图 10-4）。这是将复杂问题（例如，非线性

问题)分解成一系列较简单问题(例如，线性问题)的又一种方法。我们已经看到使用这一思想的决策树(第9章)，并且在第15章，我们还将看到组合多个模型的更多例子，例如校错输出码和混合专家模型，其中线性模型数小于$\mathcal{O}(K^2)$。

10.5 参数判别式的进一步讨论

在第5章，我们看到如果类密度$p(\boldsymbol{x} \mid C_i)$是高斯的，并且具有共同的协方差矩阵，则判别式函数是线性的

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} \tag{10.11}$$

其中参数可以用下式解析地计算

$$\begin{aligned} \boldsymbol{w}_i &= \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i \\ w_{i0} &= -\frac{1}{2}\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \log P(C_i) \end{aligned} \tag{10.12}$$

给定数据集，我们首先计算$\boldsymbol{\mu}_i$和$\boldsymbol{\Sigma}$的估计，然后把估计$\boldsymbol{m}_i$和$\mathbf{S}$插入(10.12)式，并计算线性判别式的参数。

让我们再次考虑两类的特殊情况。我们定义$y \equiv P(C_1 \mid \boldsymbol{x})$，$p(C_2 \mid \boldsymbol{x}) = 1 - y$。则在分类时，我们

$$\text{选择 } C_1\text{，如果} \begin{cases} y > 0.5 \\ \dfrac{y}{1-y} > 1, & \text{否则选择 } C_2\text{。} \\ \log \dfrac{y}{1-y} > 0 \end{cases}$$

$\log y/(1-y)$称作分对数(logit)变换或y的对数几率(log odd)。在两个共享相同的协方差矩阵的正态类的情况下，对数几率是线性的： 205

$$\begin{aligned} \text{logit}(P(C_1 \mid \boldsymbol{x})) &= \log \frac{P(C_1 \mid \boldsymbol{x})}{1 - P(C_1 \mid \boldsymbol{x})} = \log \frac{P(C_1 \mid \boldsymbol{x})}{P(C_2 \mid \boldsymbol{x})} \\ &= \log \frac{p(\boldsymbol{x} \mid C_1)}{p(\boldsymbol{x} \mid C_2)} + \log \frac{P(C_1)}{P(C_2)} \\ &= \log \frac{(2\pi)^{-d/2} \mid \boldsymbol{\Sigma} \mid^{-1/2} \exp[-(1/2)(\boldsymbol{x} - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_1)]}{(2\pi)^{-d/2} \mid \boldsymbol{\Sigma} \mid^{-1/2} \exp[-(1/2)(\boldsymbol{x} - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_2)]} + \log \frac{P(C_1)}{P(C_2)} \\ &= \boldsymbol{w}^T \boldsymbol{x} + w_0 \end{aligned} \tag{10.13}$$

其中

$$\begin{aligned} w &= \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \log \frac{P(C_1)}{P(C_2)} \end{aligned} \tag{10.14}$$

分对数的逆

$$\log \frac{P(C_1 \mid \boldsymbol{x})}{1 - p(C_1 \mid \boldsymbol{x})} = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

是逻辑斯谛(logistic)函数，又称S形(sigmoid)函数(见图10-5)：

$$P(C_1 \mid \boldsymbol{x}) = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x} + w_0) = \frac{1}{1 + \exp[-(\boldsymbol{w}^T\boldsymbol{x} + w_0)]} \tag{10.15}$$

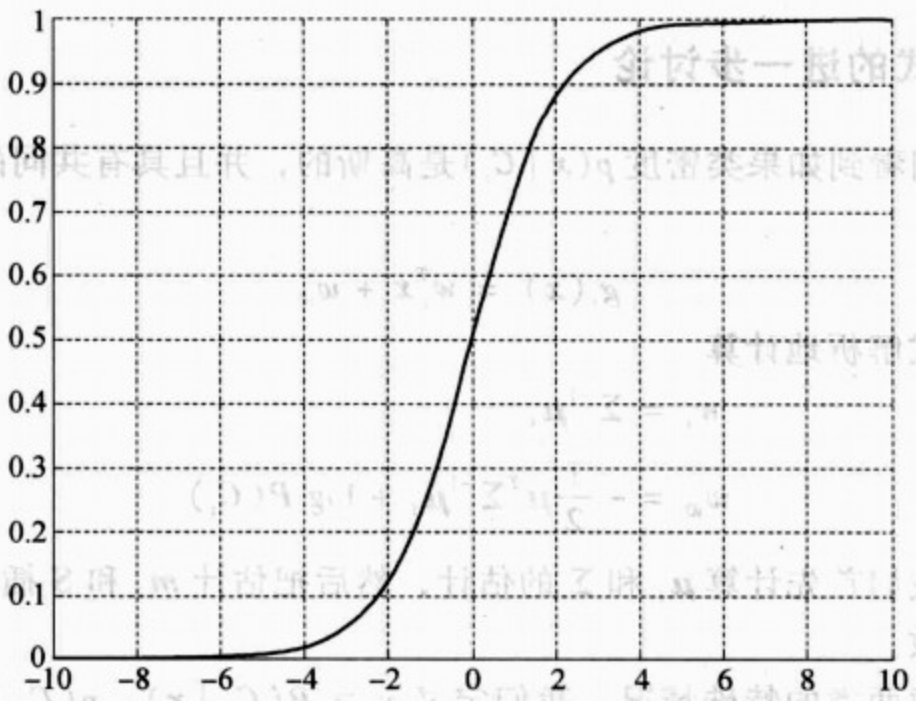


图10-5 逻辑斯谛(或S形)函数

在训练阶段，我们估计 $\boldsymbol{m}_1$，$\boldsymbol{m}_2$，$\mathbf{S}$，并将这些估计插入(10.14)式，计算判别式的参数。在检验阶段，给定 $\boldsymbol{x}$，我们可以

1. 计算 $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$，并且如果 $g(\boldsymbol{x}) > 0$，则选择 C_1；或者
2. 计算 $y = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x} + w_0)$，并且如果 $y > 0.5$，则选择 C_1，因为 $\text{sigmoid}(0) = 0.5$。

在后一种情况下，S形函数将判别式的值变换为后验概率。当有两个类并且只有一个判别式时，这是有效的。在10.7节，我们将讨论如何对 $K > 2$ 估计后验概率。

10.6 梯度下降

在基于似然的分类，参数是 $p(\boldsymbol{x} \mid C_i)$ 和 $P(C_i)$ 的有效统计量，而我们使用的估计参数的方法是最大化似然。在基于判别式的方法中，参数是判别式的参数，并且它们在最小化训练集上的分类误差是最优的。当 $\boldsymbol{w}$ 表示参数集，$E(\boldsymbol{w} \mid \mathcal{X})$ 表示参数 $\boldsymbol{w}$ 在给定训练集 $\mathcal{X}$ 上的误差时，我们寻找

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}}{\text{argmin}} E(\boldsymbol{w} \mid \mathcal{X})$$

在许多情况下，其中的一些稍后我们将看到，不存在解析解，而我们需要求助于迭代优化方法。最常用的方法是梯度下降(gradient descent)方法：当 $E(\boldsymbol{w})$ 是变量向量的可微函数时，我们有偏导数组成的梯度向量(gradient vector)

$$\nabla_{\boldsymbol{w}} E = \left[\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \cdots, \frac{\partial E}{\partial w_d}\right]^T$$

和梯度下降（gradient descent）过程来最小化 E。该方法从随机向量 $\boldsymbol{w}$ 开始，并在每一步沿与该梯度相反的方向更新 $\boldsymbol{w}$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}, \forall i \tag{10.16}$$

$$w_i = w_i + \Delta w_i \tag{10.17}$$

其中 η 称作步长（stepsize）或学习因子（learning factor），决定向该方向移动多少。梯度上升用来最大化函数，并沿着梯度的方向前进。当我们得到极小（或极大）值时，导数等于 0，过程终止。这表明过程找到了一个最近的极小值，可能是局部极小值。除非函数只有一个极小值，否则不能找到全局极小。使用较好的 η 值也是至关重要的。如果太小，收敛可能太慢；太大可能导致摆动甚至发散。

206 ~ 207

在本书中，我们使用的梯度方法很简单，并且相当有效。然而，我们要记住，一旦确定了合适的方法和误差函数，就可以使用多种可能技术中的一种来优化模型参数，以便最小化误差函数。存在一些二阶方法和共轭梯度，收敛更快，但内存开销和计算量更大。像模拟退火和遗传算法这样的开销更大的方法可以更彻底地搜索参数空间，而不太依赖初始点的选择。

10.7 逻辑斯谛判别式

10.7.1 两类问题

在*逻辑斯谛判别式*（logistic discrimination）中，我们不是对类条件密度 $p(\boldsymbol{x} \mid C_i)$，而是对它们的比率建模。让我们还是从两类问题开始，并假定对数似然比是线性的：

$$\log \frac{p(\boldsymbol{x} \mid C_1)}{p(\boldsymbol{x} \mid C_2)} = \boldsymbol{w}^T \boldsymbol{x} + w_0^o \tag{10.18}$$

当类条件密度为正态时(10.13)，这种假设确实成立。但是，逻辑斯谛判别式具有更广泛的应用。例如，$\boldsymbol{x}$ 可能由离散属性组成，或者可能是连续和离散属性的混合。

使用贝叶斯规则，我们有

$$\begin{aligned} \text{logit}(P(C_1 \mid \boldsymbol{x})) &= \log \frac{P(C_1 \mid \boldsymbol{x})}{1 - P(C_1 \mid \boldsymbol{x})} \\ &= \log \frac{p(\boldsymbol{x} \mid C_1)}{p(\boldsymbol{x} \mid C_2)} + \log \frac{P(C_1)}{P(C_2)} \\ &= w^T \boldsymbol{x} + w_0 \end{aligned} \tag{10.19}$$

208

其中

$$w_0 = w_0^o + \log \frac{P(C_1)}{P(C_2)} \tag{10.20}$$

重新整理，我们又得到 S 形函数

$$y = \hat{p}(C_1 \mid \boldsymbol{x}) = \frac{1}{1 + \exp[-(\boldsymbol{w}^T \boldsymbol{x} + w_0)]} \tag{10.21}$$

作为 $P(C_1 \mid \boldsymbol{x})$ 的估计。

让我们看看如何学习 $\boldsymbol{w}$ 和 w_0：给定两个类的样本 $\mathcal{X} = \{\boldsymbol{x}^t, r^t\}$，其中如果 $\boldsymbol{x} \in C_1$ 则 $r^t = 1$，如果 $\boldsymbol{x} \in C_2$ 则 $r^t = 0$。我们假定给定 $\boldsymbol{x}^t$，r^t 是伯努利分布，具有(10.21)式计算的概率 $y^t \equiv (P(C_1 \mid \boldsymbol{x}^t)$：

$$r^t \mid \boldsymbol{x}^t \sim \text{Bernoulli}(y^t)$$

这里，我们看到了基于似然的方法与基于判别式的方法的区别：对于前者，我们对

$p(\boldsymbol{x} \mid C_i)$建模；对于后者，我们直接对 $r \mid \boldsymbol{x}$ 建模。样本的似然是

$$l(\boldsymbol{w}, w_0 \mid \mathcal{X}) = \prod_t (y^t)^{(r^t)} (1 - y^t)^{(1-r^t)} \tag{10.22}$$

我们知道，当我们有一个需要最大化的似然函数时，我们总是将它转换成需要最小化的误差函数 $E = -\log l$，并且在我们的问题中，我们有互熵（cross-entropy）：

$$E(\boldsymbol{w}, w_0 \mid \mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t) \log(1 - y^t) \tag{10.23}$$

我们使用梯度下降法最小化互熵，等价于最大化似然或对数似然。如果 $y = \text{sigmoid}(a) = 1/(1 + \exp(-a))$，则它的导数为

$$\frac{dy}{da} = y(1 - y)$$

并且我们得到如下更新方程：

$$\begin{aligned} \Delta w_j &= -\eta \frac{\partial E}{\partial w_j} = \eta \sum_t \left(\frac{r^t}{y^t} - \frac{1 - r^t}{1 - y^t} \right) y^t (1 - y^t) x_j^t \\ &= \eta \sum_t (r^t - y^t) x_j^t, j = 1, \cdots, d \end{aligned}$$

209

$$\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_t (r^t - y^t) \tag{10.24}$$

最好用接近于 0 的随机值初始化 w_j；通常，它们从区间$[-0.01, 0.01]$中均匀地抽取。这样做的理由是，如果 w_j 数值很大，则加权和可能也很大并且 S 形函数可能饱和。从图 10-5 我们看到，如果初始权重接近于 0，则和在区域中间，那里导数非零，可以进行更新。如果加权和很大（小于 -5 或大于 +5），则 S 形函数的导数几乎为 0，权值将不会更新。

伪代码在图 10-6 中。我们看图 10-7 中的例子，其中输入是一维的。直线 $wx + w_0$ 和它的 S 形函数之后的值都作为学习迭代次数的函数显示。我们看到，为了得到输出 0 和 1，S 形函数逐渐适应，这通过增大 $\boldsymbol{w}$ 的数值实现。

```
For j = 0, ..., d
  w_j ← rand(-0.01, 0.01)
Repeat
  For j = 0, ..., d
    Δw_j ← 0
  For t = 1, ..., N
    o ← 0
    For j = 0, ..., d
      o ← o + w_j x_j^t
    y ← sigmoid(o)
  For j = 0, ..., d
    Δw_j ← Δw_j + (r^t - y) x_j^t
  For j = 0, ..., d
    w_j ← w_j + η Δw_j
Until 收敛
```

图 10-6 对于具有两个类、单个输出，实现梯度下降的逻辑斯谛判别分析算法。对于 w_0，我们假定存在一个附加的输入 x_0，它总为 +1：$x_0^t \equiv +1, \forall t$

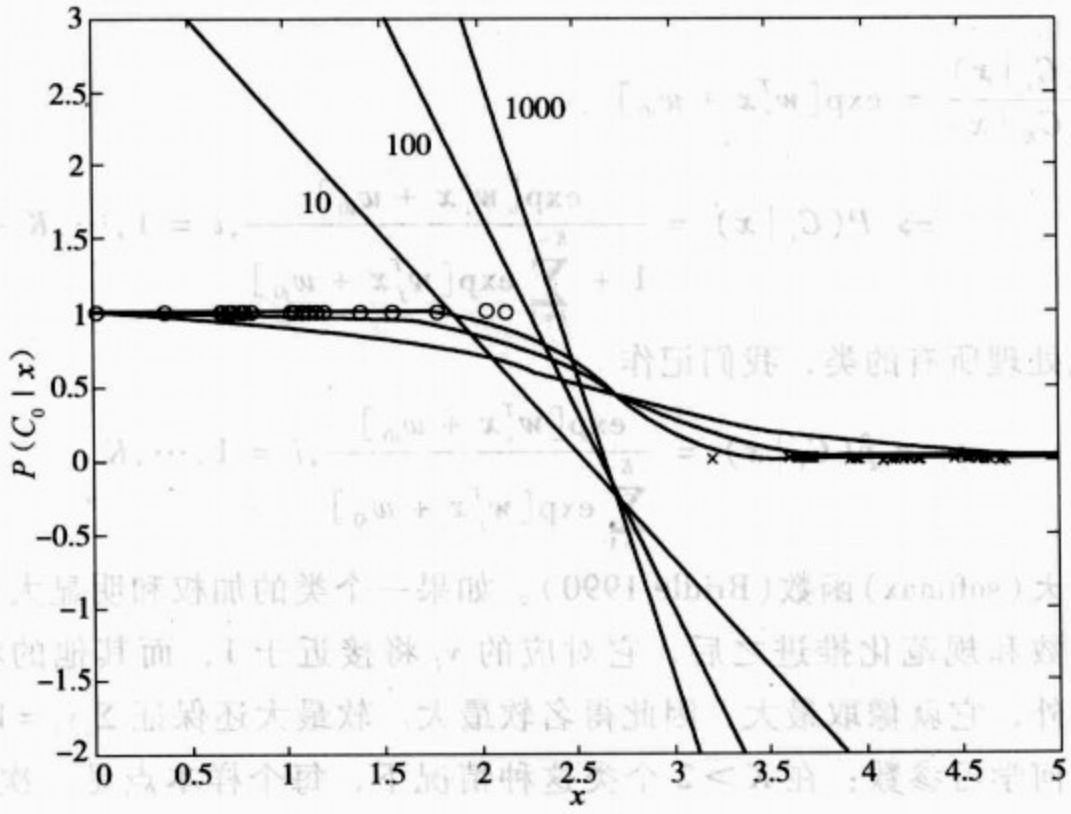


图 10-7 对于一元两类问题(用“∘”和“×”显示)，样本上 10 次、100 次和 1 000 次迭代之后，直线 $wx + w_0$ 和 S 形函数输出的演变

一旦训练完成并且我们得到了最终的 $\boldsymbol{w}$ 和 w_0，在检验阶段，给定 $\boldsymbol{x}$，我们计算 $y = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x} + w_0)$，并且如果 $y > 0.5$ 则选择 C_1，否则选择 C_2。这意味，为了最小化误分类数，我们不需要学习到 y 是 0 或 1，而只需要学习到 y 小于或大于 0.5。如果超过该点后我们还继续学习，互熵将继续降低($|w_j|$ 将继续增加，硬化 S 形函数)，但是误分类数将不会减少(如果类是线性可分的，它将为 0)。

注意，尽管为了导出判别式，我们假定类密度的对数比是线性的，但是我们直接估计后验概率，而不显式地估计 $p(\boldsymbol{x} \mid C_i)$ 或 $P(C_i)$。

10.7.2 多类问题

现在，让我们推广到 $K > 2$ 个类：我们取其中一个类，例如 C_k，作为引述类并假定

$$\log \frac{p(\boldsymbol{x} \mid C_i)}{p(\boldsymbol{x} \mid C_K)} = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}^o \tag{10.25}$$

210
≀
211

于是，我们有

$$\frac{P(C_i \mid \boldsymbol{x})}{P(C_K \mid \boldsymbol{x})} = \exp[\boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}] \tag{10.26}$$

其中 $w_{i0} = w_{i0}^0 + \log P(C_i)/P(C_k)$。

我们看到

$$\sum_{i=1}^{K-1} \frac{P(C_i \mid \boldsymbol{x})}{P(C_K \mid \boldsymbol{x})} = \frac{1 - P(C_K \mid \boldsymbol{x})}{P(C_K \mid \boldsymbol{x})} = \sum_{i=1}^{K-1} \exp[\boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}]$$

$$\Rightarrow P(C_K \mid \boldsymbol{x}) = \frac{1}{1 + \sum_{i=1}^{K-1} \exp[\boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}]} \tag{10.27}$$

并且还有

$$\frac{P(C_i \mid \boldsymbol{x})}{P(C_K \mid \boldsymbol{x})} = \exp[\boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}]$$

$$\Rightarrow P(C_i \mid \boldsymbol{x}) = \frac{\exp[\boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}]}{1 + \sum_{j=1}^{K-1} \exp[\boldsymbol{w}_j^T \boldsymbol{x} + w_{j0}]}, i = 1, \cdots, K-1 \tag{10.28}$$

为了一致地处理所有的类，我们记作

$$y_i = \hat{p}(C_i \mid \boldsymbol{x}) = \frac{\exp[\boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}]}{\sum_{j=1}^{K} \exp[\boldsymbol{w}_j^T \boldsymbol{x} + w_{j0}]}, i = 1, \cdots, K \tag{10.29}$$

这称为软最大(softmax)函数(Bridle 1990)。如果一个类的加权和明显大于其他类的加权和，则通过取指数和规范化推进之后，它对应的 y_i 将接近于1，而其他的将接近于0。这样，除了可导之外，它就像取最大，因此得名软最大。软最大还保证 $\Sigma_i y_i = 1$。

让我们看如何学习参数：在 $K>2$ 个类这种情况下，每个样本点是一次多项试验取值，即 $\boldsymbol{r}^t \mid \boldsymbol{x}^t \sim \text{Mult}_k(1, \boldsymbol{y}^t)$，其中 $y_i^t \equiv P(C_i \mid \boldsymbol{x}^t)$。样本的似然为

$$l(\{\boldsymbol{w}_i, w_{i0}\}_i \mid \mathcal{X}) = \prod_t \prod_i (y_i^t)^{r_i^t} \tag{10.30}$$

而误差函数又是互熵：

$$E(\{\boldsymbol{w}_i, w_{i0}\}_i \mid \mathcal{X}) = -\sum_t \sum_i r_i^t \log y_i^t \tag{10.31}$$

我们再次使用梯度下降。如果 $y_i = \exp(a_i)/\Sigma_j \exp(a_j)$，则我们有

212

$$\frac{\partial y_i}{\partial a_j} = y_i(\delta_{ij} - y_j) \tag{10.32}$$

其中 δ_{ij} 是 Kronecker δ，如果 $i=j$ 它为1，如果 $i \neq j$ 它为0(习题3)。给定 $\Sigma_i r_i^t = 1$，对于 $j = 1, \cdots, K$，我们有如下更新方程

$$\begin{aligned}
\Delta \boldsymbol{w}_j &= \eta \sum_t \sum_i \frac{r_i^t}{y_i^t} y_i^t (\delta_{ij} - y_j^t) \boldsymbol{x}^t \\
&= \eta \sum_t \sum_i r_i^t (\delta_{ij} - y_j^t) \boldsymbol{x}^t \\
&= \eta \sum_t \left[\sum_i r_j^t \delta_{ij} - y_j^t \sum_i r_i^t \right] \boldsymbol{x}^t \\
&= \eta \sum_t (r_j^t - y_j^t) \boldsymbol{x}^t \\
\Delta w_{j0} &= \eta \sum_t (r_j^t - y_j^t)
\end{aligned} \tag{10.33}$$

注意，由于软最大中的规范化，w_j 和 w_{j0} 不仅受 $\boldsymbol{x}^t \in C_j$ 的影响，而且还受 $\boldsymbol{x}^t \in C_i (i \neq j)$ 的影响。更新判别式使得取软最大之后正确的类具有最大的加权和，而其他类的加权和尽可能小。伪代码在图10-8中给出。对于具有三个类的二维样本，轮廓线在图10-9中给出，而判别式和后验概率在图10-10中。

For $i=1, \cdots, K$, For $j=0, \cdots, d$, $w_{ij} \leftarrow \text{rand}(-0.01, 0.01)$
Repeat
  For $i=1, \cdots, K$, For $j=0, \cdots, d$, $\Delta w_{ij} \leftarrow 0$
  For $t=1, \cdots, N$
    For $i=1, \cdots, K$
      $o_i \leftarrow 0$
      For $j=0, \cdots, d$
        $o_i \leftarrow o_i + w_{ij} x_j^t$
    For $i=1, \cdots, K$
      $y_i \leftarrow \exp(o_i) / \sum_k \exp(o_k)$
    For $i=1, \cdots, K$
      For $j=0, \cdots, d$
        $\Delta w_{ij} \leftarrow \Delta w_{ij} + (r_i^t - y_i) x_j^t$
  For $i=1, \cdots, K$
    For $j=0, \cdots, d$
      $w_{ij} \leftarrow w_{ij} + \eta \Delta w_{ij}$
Until 收敛

图 10-8　对于 $K>2$ 个类，实现梯度下降的逻辑斯谛判别式算法。为了一般起见，对于任意 t，我们取 $x_0^t \equiv 1$

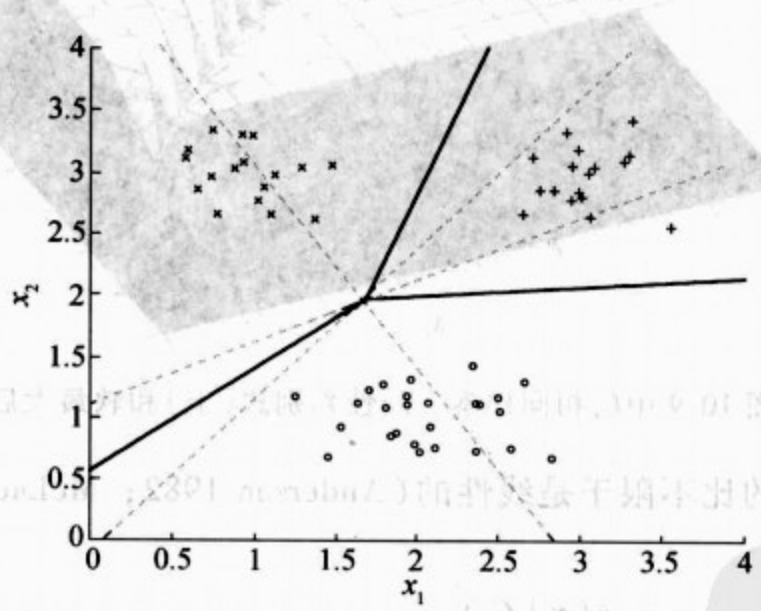


图 10-9　对于具有三个类的二维问题，逻辑斯谛判别式发现的解。细线是 $g_i(x)=0$，而粗线是取极大的线性分类器得到的边界

在检验阶段，我们计算所有的 y_k，$k=1, \cdots, K$，并且如果 $y_i = \max_k y_k$，则选择 C_i。我们仍然不必为尽可能地极小化互熵而一直训练；我们只需要训练到正确的类具有最大的加权和，并通过检查误分类数，提前停止训练。

当数据是正态分布时，逻辑斯谛判别式与参数的、基于正态的线性判别式具有大致相当的错误率(McLachlan 1992)。当类条件密度非正态时，或当它们非单峰时，只要类是线性可分的，逻辑斯谛判别式仍然可以使用。

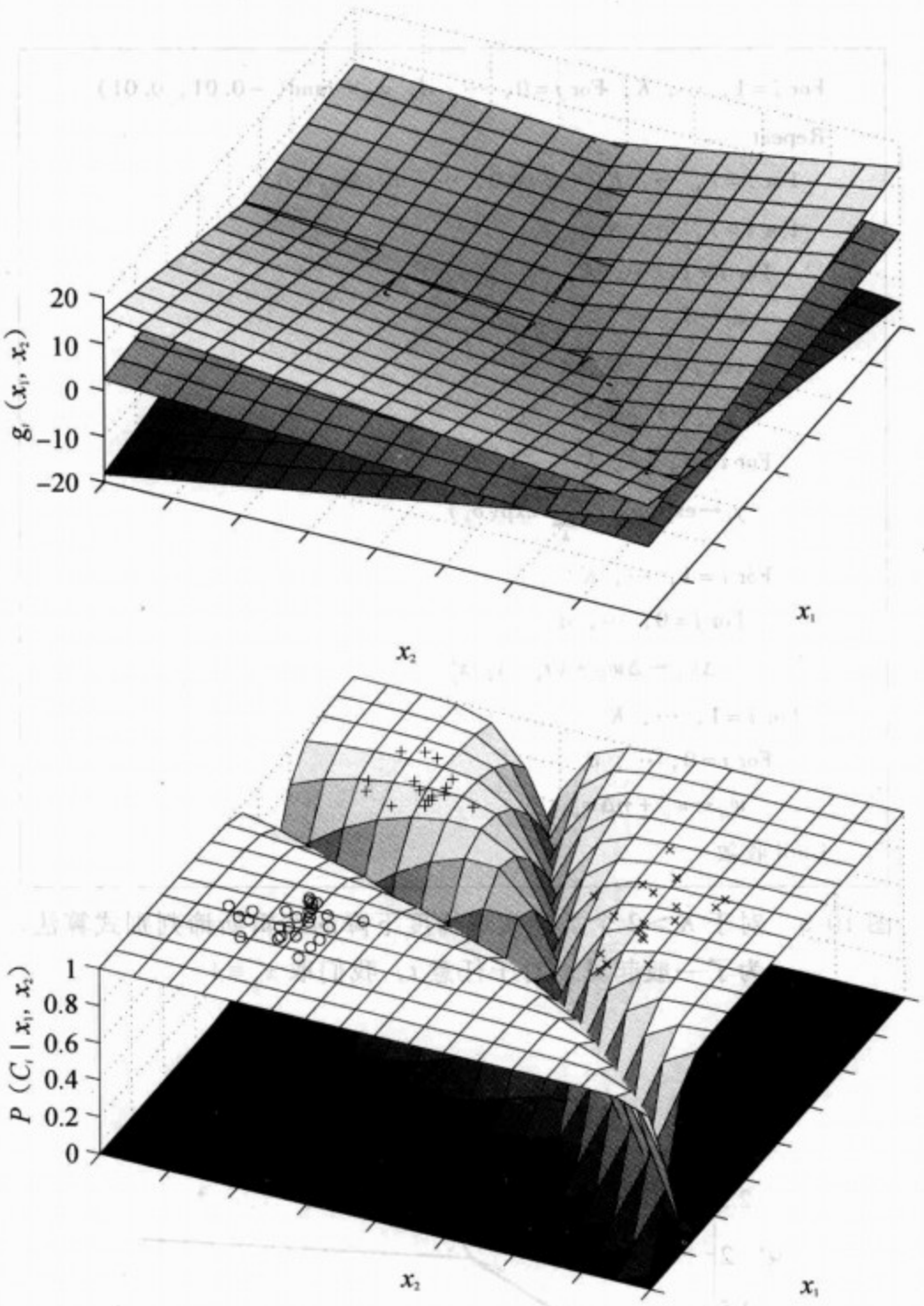


图 10-10 对于图 10-9 中的相同样本，线性判别式(上)和软最大后的后验概率(下)

当然，类条件密度的比不限于是线性的(Anderson 1982；McLachlan 1992)。假定一个二次判别式，我们有

$$\log \frac{p(\boldsymbol{x} \mid C_i)}{p(\boldsymbol{x} \mid C_k)} = \boldsymbol{x}^T \mathbf{W}_i \boldsymbol{x} + \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0} \tag{10.34}$$

对应并推广具有不同协方差矩阵、满足多元正态类条件分布的参数判别式。当 d 很大时，正如我们可以化简(正规化)$\boldsymbol{\Sigma}_i$ 一样，我们可以通过只考虑它的前面的本征向量，对 $\mathbf{W}_i$ 做同样的事。

正如 10.2 节所讨论的，可以用基本变量的任意指定函数作为 x-变量。例如，我们可以把判别式写成非线性基函数的线性和

$$\log \frac{p(\boldsymbol{x} \mid C_i)}{p(\boldsymbol{x} \mid C_k)} = \boldsymbol{w}_i^T \boldsymbol{\phi}(\boldsymbol{x}) + w_{i0} \tag{10.35}$$

其中，$\boldsymbol{\Phi}(\cdot)$ 是基函数，可以看作变换后的变量。用神经网络的术语，这称作多层感知器(multilayer perception)(第 11 章)，而 S 形函数是最常用的基函数。当使用高斯基函数时，这种模型称作径向基函数(radial basis function)(第 12 章)。我们甚至可以使用完全非参数的方法，如 Parzen 窗口(第 8 章)。

10.8 回归判别式

对于回归，概率模型是

$$r^t = y^t + \varepsilon \tag{10.36}$$

其中 $\varepsilon \sim \mathcal{N}(0, \sigma^2)$。如果 $r^t \in \{0, 1\}$，使用 S 形函数，则 y^t 可能限于落在该区间。假定线性模型和两个类，我们有

$$y^t = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) = \frac{1}{1 + \exp[-(\boldsymbol{w}^T\boldsymbol{x}^t + w_0)]} \tag{10.37}$$

假定 $r \mid \boldsymbol{x} \sim \mathcal{N}(y, \sigma^2)$，则回归中的样本似然为

$$l(\boldsymbol{w}, w_0 \mid \mathcal{X}) = \prod_t \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(r^t - y^t)^2}{2\sigma^2}\right] \tag{10.38}$$

最大化该对数似然是最小化误差的平方和：

$$E(\boldsymbol{w}, w_0 \mid \mathcal{X}) = \frac{1}{2}\sum_t (r^t - y^t)^2 \tag{10.39}$$

使用梯度下降，我们得到

$$\Delta \boldsymbol{w} = \eta \sum_t (r^t - y^t) y^t (1 - y^t) \boldsymbol{x}^t$$

$$\Delta w_0 = \eta \sum_t (r^t - y^t) y^t (1 - y^t) \tag{10.40}$$

213 ~ 216

当存在 $K > 2$ 个类时，也可以使用这种方法。概率模型是

$$\boldsymbol{r}^t = \boldsymbol{y}^t + \boldsymbol{\varepsilon} \tag{10.41}$$

其中 $\boldsymbol{\varepsilon} \sim \mathcal{N}_k(0, \sigma^2 \mathbf{I}_K)$。假定每个类一个线性模型，我们有

$$y_i^t = \text{sigmoid}(\boldsymbol{w}_i^T\boldsymbol{x}^t + w_{i0}) = \frac{1}{1 + \exp[-(\boldsymbol{w}_i^T\boldsymbol{x}^t + w_{i0})]} \tag{10.42}$$

于是，样本的似然为

$$l(\{\boldsymbol{w}_i, w_{i0}\}_i \mid \mathcal{X}) = \prod_t \frac{1}{(2\pi)^{K/2} \mid \Sigma \mid^{1/2}} \exp\left[-\frac{\| \boldsymbol{r}^t - \boldsymbol{y}^t \|^2}{2\sigma^2}\right] \tag{10.43}$$

而误差函数为

$$E(\{\boldsymbol{w}_i, w_{i0}\}_i \mid \mathcal{X}) = \frac{1}{2}\sum_t \| \boldsymbol{r}^t - \boldsymbol{y}^t \|^2 = \frac{1}{2}\sum_t \sum_i (r_i^t - y_i^t)^2 \tag{10.44}$$

对于 $i = 1, \cdots, K$，更新方程为

$$\Delta \boldsymbol{w}_i = \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t) \boldsymbol{x}^t$$

$$\Delta w_{i0} = \eta \sum_t (r_i^t - y_i^t) y_i^t (1 - y_i^t) \tag{10.45}$$

注意，在这样做时，我们并未使用 y_i 中的一个为 1，其余为 0，或 $\sum_i y_i = 1$ 的信息。由

于输出是类后验概率估计，(10.29)式的软最大函数使得我们可以纳入这些附加信息。在$K>2$的情况下，使用S形函数，我们像对待独立函数一样对待y_i。

还要注意，对于给定的类，如果我们使用回归方法，则更新将进行到正确的输出为1，其余为0才停止。事实上，这是不必要的，因为在检验时，我们只是选择最大的。训练到正确的输出大于其他输出就足够了，这正是软最大函数所做的。

因此，当类不是互斥的和穷举的时，也就是说，对于一个$\boldsymbol{x}^t$，所有的r_i^t可能都为0，即$\boldsymbol{x}^t$不属于任何一个类；或者当类重叠时，可能多个r_i^t为1时，这种具有多个S形函数的方法更可取。

217

10.9 支持向量机

10.9.1 最佳分离超平面

现在，我们讨论一种学习线性判别式的不同方法。我们不必惊奇，即使对于线性分类这种简单情况，也存在许多不同的方法。每种方法都具有不同的归纳偏倚，做不同的假设，定义不同的目标函数，因此可能发现不同的线性判别式。

让我们还是从两类开始，并使用$-1/+1$标记这两个类。样本为$\mathcal{X}=\{\boldsymbol{x}^t, r^t\}$，其中如果$\boldsymbol{x}^t\in C_1$则$r^t=+1$，如果$\boldsymbol{x}^t\in C_2$则$r^t=-1$。我们希望找到$\boldsymbol{w}$和$w_0$，使得

$$\text{对于 } r^t=+1, \quad \boldsymbol{w}^T\boldsymbol{x}^t+w_0\geq+1$$

$$\text{对于 } r^t=-1, \quad \boldsymbol{w}^T\boldsymbol{x}^t+w_0\leq-1$$

它们可以合并，改写为

$$r^t(\boldsymbol{w}^T\boldsymbol{x}^t+w_0)\geq+1 \tag{10.46}$$

注意，我们并不是简单地要求

$$r^t(\boldsymbol{w}^T\boldsymbol{x}^t+w_0)\geq 0$$

为了更好地泛化，我们不仅希望实例在超平面的正确的一侧，而且我们还希望它们离超平面有一定距离。超平面到它两侧最近实例的距离称作边缘(margin)。为了更好地泛化，我们希望最大化边缘。最佳分离超平面(optimal separating hyperplane)是最大化边缘的超平面。

回忆10.3节，$\boldsymbol{x}^t$到判别式的距离为

$$\frac{|\boldsymbol{w}^T\boldsymbol{x}^t+w_0|}{\|\boldsymbol{w}\|}$$

当$r^t\in\{-1, +1\}$时，上式可以记作

$$\frac{r^t(\boldsymbol{w}^T\boldsymbol{x}^t+w_0)}{\|\boldsymbol{w}\|}$$

至少对于某个ρ，我们希望它

$$\frac{r^t(\boldsymbol{w}^T\boldsymbol{x}^t+w_0)}{\|\boldsymbol{w}\|}\geq\rho, \forall t \tag{10.47}$$

我们希望最大化ρ，但是缩放$\boldsymbol{w}$，我们可以得到的解有无限多个。为了得到唯一的解，我们固定$\rho\|\boldsymbol{w}\|=1$。这样，为最大化边缘，我们最小化$\|\boldsymbol{w}\|$。这个任务可以定义为(见Cortes和Vapnik 1995; Vapnik 1995)：

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2, \text{受限于} r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) \geq +1, \forall t \tag{10.48}$$

这是一个标准的二次优化问题，其复杂度依赖于 d，并且可以直接求解，找到 $\boldsymbol{w}$ 和 w_0。于是，在超平面的两侧，实例离超平面至少为 $1/\|\boldsymbol{w}\|$，而整个边缘为 $2/\|\boldsymbol{w}\|$（参见图 10-11）。

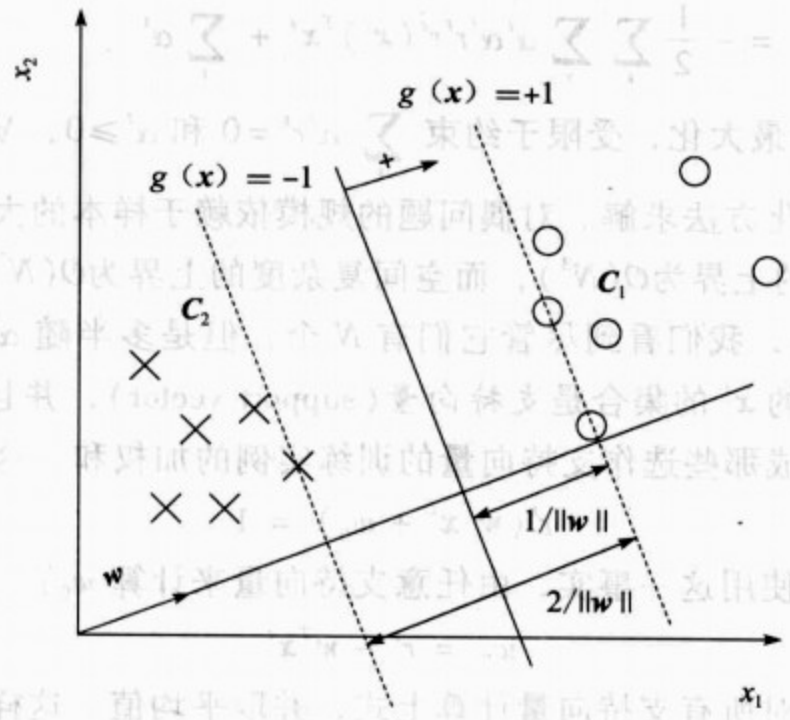


图 10-11 在最佳分离超平面两侧，实例离超平面至少为 $1/\|\boldsymbol{w}\|$，而整个边缘为 $2/\|\boldsymbol{w}\|$

在 10.2 节我们看到，如果问题不是线性可分的，我们不是拟合非线性函数，而是使用非线性基函数将问题映射到新的空间。通常，新空间的维度比原空间更高，并且在这种情况下，我们对复杂度不依赖输入维度的方法感兴趣。

218
≀
219

在找最佳分离超平面时，我们可以把该优化问题转换成复杂度依赖于训练实例数 N，而不依赖于 d 的形式。这种新公式的另一个优点是它使得我们可以用核函数改写基函数，如 10.9.3 节所示。

为了得到新的公式，我们使用拉格朗日乘子 α^t，将(10.48)式改写成非约束问题：

$$\begin{aligned} L_p &= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{t=1}^{N} \alpha^t [r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) - 1] \\ &= \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_t \alpha^t r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) + \sum_t \alpha^t \end{aligned} \tag{10.49}$$

这应当关于 $\boldsymbol{w}$，w_0 最小化，关于 $\alpha^t \geq 0$ 最大化。鞍点给出解。

这是一个凸二次优化问题，因为主要项是凸的，并且线性约束也是凸的。这样，我们可以使用 Karush-Kuhn-Tucker 条件，解其对偶问题。对偶问题是关于 α^t 最大化 L_p，受限于约束 L_p 关于 $\boldsymbol{w}$ 和 w_0 的梯度为 0，并且 $\alpha^t \geq 0$：

$$\frac{\partial L_p}{\partial \boldsymbol{w}} = 0 \Rightarrow \boldsymbol{w} = \sum_t \alpha^t r^t \boldsymbol{x}^t \tag{10.50}$$

$$\frac{\partial L_p}{\partial w_0} = 0 \Rightarrow \sum_t \alpha^t r^t = 0 \tag{10.51}$$

将它们代入(10.49)式，我们得到对偶问题

$$L_d = \frac{1}{2}(\boldsymbol{w}^T\boldsymbol{w}) - \boldsymbol{w}^T\sum_t \alpha^t r^t \boldsymbol{x}^t - w_0\sum_t \alpha^t r^t + \sum_t \alpha^t$$

$$= -\frac{1}{2}(\boldsymbol{w}^T\boldsymbol{w}) + \sum_t \alpha^t$$

$$= -\frac{1}{2}\sum_t\sum_s \alpha^t\alpha^s r^t r^s (\boldsymbol{x}^t)^T\boldsymbol{x}^s + \sum_t \alpha^t \tag{10.52}$$

我们只需要关于 α^t 对它最大化，受限于约束 $\sum_t \alpha^t r^t = 0$ 和 $\alpha^t \geq 0$，$\forall t$。

这可以使用二次优化方法求解。对偶问题的规模依赖于样本的大小 N，而不依赖于输入
220 的维度 d。时间复杂度的上界为 $\mathcal{O}(N^3)$，而空间复杂度的上界为 $\mathcal{O}(N^2)$。

一旦我们对 α^t 求解，我们看到尽管它们有 N 个，但是多半随 $\alpha^t = 0$ 消失，而只有少量满足 $\alpha^t > 0$。其 $\alpha^t > 0$ 的 $\boldsymbol{x}^t$ 的集合是支持向量(support vector)，并且正如我们在(10.50)式中所看到的，$\boldsymbol{w}$ 可以写成那些选作支持向量的训练实例的加权和。这些 $\boldsymbol{x}^t$ 满足

$$r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) = 1$$

并且落在边缘上。我们使用这一事实，由任意支持向量来计算 w_0：

$$w_0 = r^t - \boldsymbol{w}^T\boldsymbol{x}^t \tag{10.53}$$

从稳定性讲，建议对所有支持向量计算上式，并取平均值。这样找出的判别式称作支持向量机(support vector machine，SVM)。

α^t 大部分为0，对于它们，$r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) > 1$。这些 $\boldsymbol{x}^t$ 落在远离边缘内部的地方，并且对超平面没有影响。从这个角度讲，这种算法与精简的最近邻算法(8.5节)类似，它只保存定义类判别式的实例。作为基于判别式的算法，SVM 只关注那些靠近边界的实例，而丢弃那些落在内部的实例。使用这种思想，可以在求 SVM 之前先使用一种较简单的分类器过滤掉这种实例的大部分，从而降低 SVM 优化阶段的复杂度。

在检验阶段，我们不强调边缘。我们计算 $g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$，并根据 $g(\boldsymbol{x})$ 的符号选择：

如果 $g(\boldsymbol{x}) > 0$ 选择 C_1，否则选择 C_2。

当存在 $K > 2$ 个类时，一种直截了当的方法是定义 K 个两类问题，每个将一个类与其他类分开，并学习 K 个支持向量机 $g_i(\boldsymbol{x})$，$i = 1, \cdots, K$。在检验阶段，我们计算所有的 $g_i(\boldsymbol{x})$，并选择最大的。

10.9.2 不可分情况：软边缘超平面

如果数据不是线性可分的，则我们前面讨论的算法就不能解决问题。在这种情况下，如
221 果两个类不是线性可分的，致使不存在将它们分开的超平面，则我们寻找出错最少的超平面。我们定义松弛变量(slack variable) $\xi^t \geq 0$，存放到边缘的离差。有两种类型的离差：一个实例可能位于超平面的错误一侧，并被错误地分类；或者实例可能在正确的一侧但可能落在边缘中，即离超平面不够远。放宽(10.46)式，我们要求

$$r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) \geq 1 - \xi^t \tag{10.54}$$

如果 $\xi^t = 0$，则 $\boldsymbol{x}^t$ 没有问题。如果 $0 < \xi^t < 1$，则 $\boldsymbol{x}^t$ 被正确分类，但是它在边缘中。如果 $\xi^t \geq 1$，则 $\boldsymbol{x}^t$ 被错误地分类(见图10-12)。误分类数为 $\#\{\xi^t \geq 1\}$，并且不可分的点数为

$\#\{\xi^t > 0\}$[⊖]。我们定义软误差(soft error)为 $\sum_t \xi^t$，并且将它作为罚项添加到原来的(10.49)式中：

$$L_p = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_t \xi^t - \sum_t \alpha^t [r^t(\boldsymbol{w}^T\boldsymbol{x}^t + w_0) - 1 + \xi^t] - \sum_t \mu^t \xi^t \quad (10.55)$$

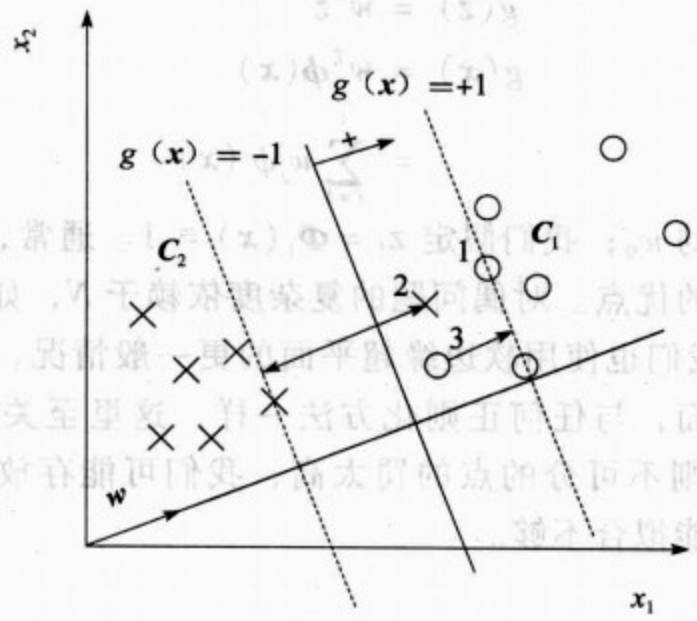


图 10-12 在对实例进行分类时，有三种可能的情况：(1)$\xi = 0$，点在正确的一侧，并且离超平面足够远。(2)$\xi = 1 + g(x) > 1$，点在错误的一侧。(3)$\xi = 1 - g(x)$，$0 < \xi < 1$，点在正确的一侧，但在边缘中，离超平面不够远

其中 μ_t 是新的拉格朗日参数，确保 ξ^t 为正。C 是罚因子，像任意正则化模式一样，在复杂度(支持向量数)和数据误拟合(不可分点数)之间权衡。注意，我们不仅罚误分类的点，为了更好的泛化也罚边缘中的点，尽管后者在检验时将正确地分类。 222

对偶问题是

$$L_d = \sum_t \alpha^t - \frac{1}{2}\sum_t \sum_s \alpha^t \alpha^s r^t r^s (\boldsymbol{x}^t)^T \boldsymbol{x}^s \quad (10.56)$$

受限于

$$\sum_t \alpha^t r^t = 0 \text{ 并且 } 0 \leqslant \alpha^t \leqslant C, \forall t$$

与可分情况一样，非支持向量的那些实例随 $\alpha^t = 0$ 消失，而其余的定义 $\boldsymbol{w}$。然后，w_0 用类似的方法求解。

10.9.3 核函数

10.2 节指出，如果问题是非线性的，我们不是训练一个非线性模型，而是使用合适的基函数，通过非线性变换将问题映射到新空间，然后再在新空间使用线性模型。新空间中的线性模型对应原空间中的非线性模型。这种方法可以用于分类和回归，并且对于分类这种特殊情况，它可以与任意模式一起使用。在支持向量机这种特定情况下，正如我们将看到的，

⊖ 这里，误分类的实例是不能用足够宽的边缘将其分开的实例。尽管其中某些被正确地分类，但是我们仍然把它们算作误分类，因为我们希望不同类的实例相距足够远。——译者注

这导致某种简化。

设我们有用基函数

$$z = \phi(x), \text{其中 } z_j = \phi_j(x), j = 1, \cdots, k$$

计算的新的维，将 d-维 x 空间映射到 k-维 z 空间，其中判别式表示为

$$\begin{aligned} g(z) &= w^T z \\ g(x) &= w^T \phi(x) \\ &= \sum_{j=1}^{k} w_j \phi_j(x) \end{aligned} \tag{10.57}$$

223

这里，我们并不使用单独的 w_0；我们假定 $z_1 = \Phi_1(x) \equiv 1$。通常，k 远大于 d，k 也大于 N，并且这也是使用对偶形式的优点。对偶问题的复杂度依赖于 N，如果我们使用原问题，则复杂度将依赖于 k。这里，我们也使用软边缘超平面的更一般情况，因为我们不能保证问题在新空间是线性可分的。然而，与任何正则化方法一样，这里至关重要的是为罚因子 C 选取适当的值。如果 C 太大，则不可分的点的罚太高，我们可能存放很多支持向量，并可能过分拟合。如果太小，则可能拟合不够。

解是

$$w = \sum_t \alpha^t r^t z^t = \sum_t \alpha^t r^t \phi(x^t) \tag{10.58}$$

而判别式是

$$g(x) = w^T \phi(x) = \sum_t \alpha^t r^t \phi(x^t)^T \phi(x) \tag{10.59}$$

核机器(kernel machine)的基本思想是用支持向量和原输入空间中的输入之间的核函数(kernel function)$K(x^t, x)$取代基函数的内积 $\phi(x^t)^T \phi(x)$：

$$g(x) = \sum_t \alpha^t r^t K(x^t, x) \tag{10.60}$$

最流行的核函数有

- q 次多项式：

$$K(x^t, x) = (x^T x^t + 1)^q$$

其中 q 由用户选择。例如，当 $q=2$，$d=2$ 时，

$$\begin{aligned} K(x, y) &= (x^T y + 1)^2 \\ &= (x_1 y_1 + x_2 y_2 + 1)^2 \\ &= 1 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 x_2 y_1 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2 \end{aligned}$$

它对应于基函数的内积(Cherkassky 和 Mulier 1998)：

224

$$\phi(x) = [1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2]^T$$

- 径向基函数：

$$K(x^t, x) = \exp\left[-\frac{\|x^t - x\|^2}{\sigma^2}\right]$$

与 Parzen 窗口一样(第 8 章)，它定义球形核，其中 x^t 是中心，而 σ 由用户提供，定义半径。这类似于第 12 章讨论的径向基函数。

- S 形函数：

$$K(x^t, x) = \tanh(2x^T x^t + 1)$$

其中 tanh(·)与 S 形函数具有相同的形状，不同的是它的取值在 -1 和 +1 之间。这类似于我们将在第 11 章讨论的多层感知器。

也可以使用其他核函数，受限于一定条件(Vapnik 1995；Cherkassky 和 Mulier 1998)。

Cortes 和 Vapnik(1995)报告了 SVM 在手写数字识别应用方面的成功结果。输入是 16 × 16 的位图，因此为 256 维。在这种情况下，使用三次多项式核意味特征空间为 10^6 维。结果表明在 7300 个实例的训练集上没有发生过分拟合，平均选取了 148 个实例作为支持向量。

Vapnik(1995)证明期望误差率为

$$E_N[P(error)] \leqslant \frac{E_N[\text{支持向量数}]}{N}$$

其中 $E_N[\cdot]$表示在大小为 N 的训练集上的期望。因此，误差率依赖于支持向量数，而不依赖输入的维度。

10.9.4 用于回归的支持向量机

尽管本章讨论分类，但是简略地讨论如何将支持向量机推广到回归也是有益的。我们使用线性模型：

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0$$

225

对于回归，我们使用差的平方作为误差：

$$e_2(r^t, f(\boldsymbol{x}^t)) = [r^t - f(\boldsymbol{x}^t)]^2$$

然而，对于支持向量回归，我们使用 ε - 敏感损失函数：

$$e_\varepsilon(r^t, f(\boldsymbol{x}^t)) = \begin{cases} 0 & \text{如果 } |r^t - f(\boldsymbol{x}^t)| < \varepsilon \\ |r^t - f(\boldsymbol{x}^t)| - \varepsilon & \text{否则} \end{cases} \tag{10.61}$$

这意味我们容忍高达 ε 的误差，并且超出的误差具有线性而不是平方影响。因此，这种误差函数更能抵御噪声，因而更加鲁棒(参见图 10-13)。

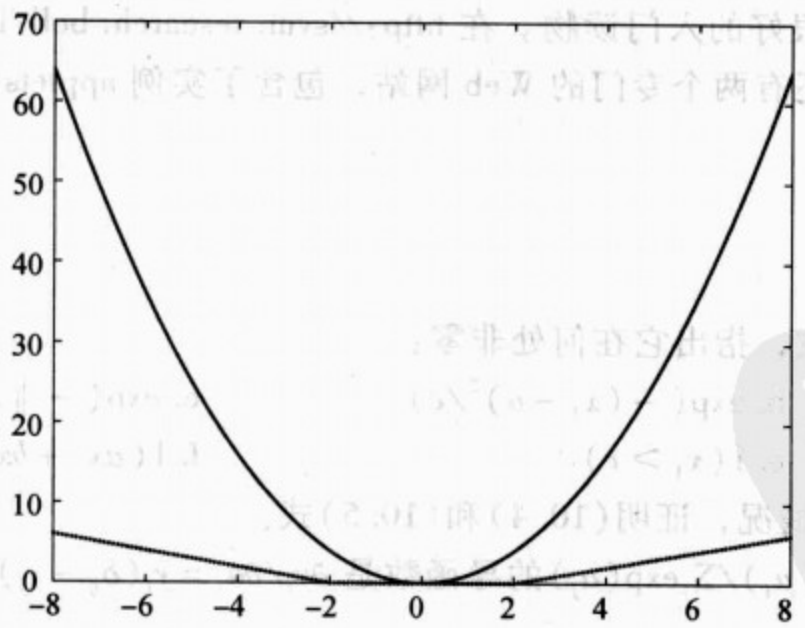


图 10-13 平方误差函数和 ε - 敏感误差函数。我们看到 ε - 敏感误差函数不受小误差的影响，并且受大误差的影响也较小，因此对离群点更鲁棒

类似于软边缘超平面，我们引入松弛变量来处理超越 ε - 区域的偏差，并且得到(Vapnik 1995)

$$\min \frac{1}{2} \| \boldsymbol{w} \|^2 + C \sum_t (\xi_+^t + \xi_-^t) \tag{10.62}$$

受限于

$$r^t - (\boldsymbol{w}^T \boldsymbol{x} + w_0) \leqslant \varepsilon + \xi_+^t$$

$$(\boldsymbol{w}^T \boldsymbol{x} + w_0) - r^t \leqslant \varepsilon + \xi_-^t$$

$$\xi_+^t, \xi_-^t \geqslant 0$$

226

这里，对正的和负的偏差，我们使用了两种类型的松弛变量，以保证它们为正。该式对应(10.61)式中给出的 ε - 敏感损失函数。

正如 Vapnik(1995)给出的，可以将它写成拉格朗日函数，然后取它的对偶问题。这里也可以使用核函数。与分类一样，结果将选择一些训练实例作为支持向量，而回归线将用它们的加权和表示。

10.10 注释

由于其简单性，线性判别分析是模式识别研究最多的课题(Duda、Hart 和 Stork 2001；McLachlan 1992)。我们在第4章讨论了具有公共协方差矩阵的高斯分布情况，在第6章讨论了费希尔线性判别式，并在本章纵览了其他方法，直到最近的支持向量机方法。在第11章，我们将讨论感知器，它是线性判别式的神经网络实现。

逻辑斯谛判别式的更详细讨论在 Anderson 1982 和 McLachlan 1992 中。逻辑斯谛(S形)函数是分对数的逆，在伯努利抽样中称作规范链(canonical link)。软最大是对多元正态抽样的拓广。关于广义线性模型(generalized linear model)的更多信息在 McCullogh 和 Nelder 1989 中。

关于支持向量机的更多信息可以在 Vapnik 所写的书(1995；1998)中找到。Cherkassky 和 Mulier 1998 中关于 SVM 的一章非常容易阅读。Burges 1998，Smola 和 Schölkopf 1998 分别是 SVM 分类和回归的很好的入门读物。在 http://svm.research.bell-labs.com 和 http://www.kernel-machines.org，还有两个专门的 Web 网站，包含了实例 applets(Java 程序)和关于 SVM 的指南和文章的链接。

10.11 习题

1. 对于如下每个基函数，指出它在何处非零：

227

a. $\sin(x_1)$	b. $\exp(-(x_1 - a)^2/c)$	c. $\exp(-\| \boldsymbol{x} - \boldsymbol{a} \|^2/c)$
d. $\log(x_2)$	e. $1(x_1 > c)$	f. $1(ax_1 + bx_2 > c)$

2. 对于图 10-2 的二维情况，证明(10.4)和(10.5)式。
3. 证明软最大 $y_i = \exp(a_i)/\Sigma_j \exp(a_j)$ 的导函数是 $\partial y_i/\partial a_i = y_i(\delta_{ij} - y_j)$，其中如果 $i = j$ 则 δ_{ij} 为1，否则 δ_{ij} 为0。
4. 令 $K = 2$，证明用两个软最大输出等于使用一个 S 形输出。
5. 在(10.34)式中，我们如何学习 $\mathbf{W}_i$?

10.12 参考文献

Aizerman, M. A., E. M. Braverman, and L. I. Rozonoer. 1964. "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning." *Automation and Remote Control* 25: 821-837.

Anderson, J. A. 1982. "Logistic Discrimination." In *Handbook of Statistics*, Vol. 2, *Classification, Pattern Recognition and Reduction of Dimensionality*, ed. P. R. Krishnaiah, L. N. Kanal, 169-191. Amsterdam: North Holland.

Bridle, J. S. 1990. "Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition." In *Neurocomputing: Algorithms, Architectures and Applications*, ed. F. Fogelman-Soulie, J. Herault, 227-236. Berlin: Springer.

Burges, C. J. C. 1998. "A Tutorial on Support Vector Machines for Pattern Recognition." *Data Mining and Knowledge Discovery* 2: 121-167.

Cherkassky, V., and F. Mulier. 1998. *Learning from Data: Concepts, Theory, and Methods*. New York: Wiley.

Cortes, C., and V. Vapnik. 1995. "Support Vector Networks." *Machine Learning* 20: 273-297.

Duda, R. O., P. E. Hart, and D. G. Stork. 2001. *Pattern Classification*, 2nd ed. New York: Wiley.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. London: Chapman and Hall.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley.

Smola, A., and B. Schölkopf. 1998. *A Tutorial on Support Vector Regression*, NeuroCOLT TR-1998-030, Royal Holloway College, University of London, UK.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. New York: Springer.

Vapnik, V. 1998. *Statistical Learning Theory*. New York: Wiley.

第11章　多层感知器

多层感知器是一种人工神经网络结构，是非参数估计器，可以用于分类和回归。我们讨论为各种应用训练多层感知器的后向传播算法。

11.1　引言

人工神经网络模型，其中之一是我们本章讨论的多层感知器，其灵感源于模拟人脑。旨在理解人脑功能，并朝着这一目标努力的认知科学家和神经学家(Posner 1989)构建了人脑的神经网络模型，并开展了模拟研究。

然而，在工程上，我们的目标不是理解人脑的本质，而是构建有用的机器。我们对人工神经网络(artificial neural networks)感兴趣，因为我们相信它们可能帮助我们建立更好的计算机系统。人脑是一种信息处理装置，具有非凡的能力并且在许多领域，例如，视觉、语音识别和学习方面，都超过了当前的工程产品。如果在机器上实现，这些应用显然都具有经济效益。如果我们能够理解人脑如何实现这些功能，我们就可以用形式算法定义这些任务的解，并且在计算机上实现它们。

人脑与计算机很不相同。计算机通常只有一个处理器，而人脑却包含大量(10^{11}个)并行

操作的处理单元，称作神经元(neuron)。尽管处理细节尚不清楚，但是人们相信这些处理单元比计算机中的处理器简单得多、并且慢得多。使得人脑不同寻常并且被认为提供了其计算能力的是连通性：人脑的神经元具有连接，称作突触(synapse)，连接到大约 10^4 个其他神经元，所有神经元都并行地操作。在计算机中，处理器是主动的，而存储是分散和被动的，但是我们认为在人脑中，处理和存储都在网络上分布；处理由神经元来做，而记忆在神经元之间的突触中。

11.1.1　理解人脑

根据 Marr(1982)，理解一个信息处理系统具有三个层面，称作分析层面(levels of analysis)：

1. 计算理论(computational theory)对应计算目标和任务的抽象定义。

2. 表示和算法(representation and algorithm)是关于输入和输出如何表示和从输入到输出变换的算法说明。

3. 硬件实现(hardware implementation)是系统的实际物理实现。

一个例子是排序：计算理论是对给定的元素集合排序。表示可以使用整数，而算法可以是 Quicksort(快速排序)。编译后，特定处理机对二进制表示的整数排序的可执行代码是一种硬件实现。

基本思想是，对于相同的计算理论，可以有多种表示和在相应表示上操控符号上的算

法。类似地，对于给定的表示和算法，可以有多种硬件实现。我们可以使用众多排序算法中的一种，并且即使相同的算法也可以在使用不同处理器的计算机上编译，导致不同的硬件实现。

考虑另一个例子，“6”、“VI”和“110”是数字6的不同表示。加法的不同算法依赖于所使用的表示。数字计算机使用二进制表示，并具有这种表示的加法电路，这是一种特殊的 230
硬件实现。在算盘上，数用不同的方法表示，并且加法对应不同的指令集，这是另一种硬件实现。当我们在大脑中将两个数相加时，我们使用另一种表示和一种适合于这种表示的算法，这由神经元实现。但是，所有不同的硬件实现(例如，我们、算盘和数字计算机)都实现了相同的计算理论——加法。

经典的例子是自然和人工的飞行器之间的不同：麻雀拍打它的双翼；商用飞机并不拍打机翼，而是使用喷气引擎。麻雀和飞机是两种硬件实现，为不同的目的而构建，满足不同的约束。但是它们都实现了相同的理论——空气动力学。

人脑是学习或模式识别的一种硬件实现。如果从这种特定的实现，我们可以做逆工程，提取人脑使用的表示和算法，并且如果我们能够从中获得计算理论，则我们可以使用另一种表示和算法，然后得到更适合我们的含义和约束的硬件实现。我们希望我们的实现价格低廉、快速、并且更准确。

当初构建飞行器时，直到发现空气动力学之前，我们一直在尝试构建看上去非常像鸟的飞行器。与此相同，直到我们发现智能的计算理论之前，早期尝试构建具有大脑能力的结构看上去将很像大脑，是具有大量处理单元的网络。因此可以说，就理解大脑而言，当我们研究人工神经网络时，我们处于表示和算法层面。

正如羽毛与飞行不相关一样，将来我们可能发现神经元和突触与智能并无关系。但是，在此之前，我们对理解大脑机能感兴趣还有另一个原因，这种原因与并行处理有关。

11.1.2 神经网络作为并行处理的典范

自20世纪80年代以来，具有数以千计处理器的计算机系统已经商品化。然而，用于这种并行结构的软件并不像硬件发展这么快。原因是到目前为止我们的计算理论几乎都基于串 231
行的、单处理器机器。我们不能有效地使用并行机，因为我们不能有效地对它们编程。

主要有两种并行处理(parallel processing)范型：在单指令多数据(SIMD)机，所有的处理器都执行相同的指令，但是在不同的数据上执行。在多指令多数据(MIMD)机，不同的处理器可以在不同的数据上执行不同的指令。SIMD机容易编程，因为只需要写一个程序。然而，问题很少具有这种有规律的结构以能在SIMD机上并行地执行。MIMD机更一般，但是为每个处理器编写单独的程序并不是一件容易的任务；其他问题涉及同步、处理器之间的数据传送等。SIMD机也比较容易构建，并且如果它们都是SIMD机，则可以构建具有更多处理器的机器。在MIMD机中，处理器更加复杂，并且还要为处理器任意地交换数据构建更复杂的通信网络。

现在，假设我们可以有机器，其中处理器比SIMD处理器复杂一点，但没有MIMD处理器复杂。假定我们有一些简单处理器，具有少量局部存储器，可以存放一些参数。每个处理器实现一个固定的函数，并且执行与SIMD处理器一样的指令；但是通过将不同的值装入局

部存储器，它们可以做不同的事情，并且整个操作可以在这些处理器上分布执行。这样，我们将有可以称作神经指令多数据(NIMD)机，其中每个处理器对应一个神经元，局部参数对应它的突触权重，而整个结构是一个神经网络。如果每个处理器中实现的功能很简单，并且局部存储器很小，则许多这样的处理器可以放在一个芯片中。

现在的问题是将任务分布到这种处理器的网络中和确定局部参数的值。这是学习进行的地方：如果这样的机器可以从实例学习，则我们自己不需要为这种机器编制程序和决定参数值。

因此，人工神经网络是一种我们可以使用当前技术构建的、利用并行硬件的方法——多亏了学习——它们不需要编程。因此，我们也不必费神为它们编程。

本章，我们讨论这种结构和如何训练它们。记住，人工神经网络操作是一种数学函数，它们可以在串行计算机上实现，并且训练网络与我们在前面的章节中讨论的统计学技术并无太大差别。仅当我们有并行硬件，并且仅当网络太大，不能在串行机上快速模拟时，考虑这些操作在简单处理单元的网络上进行才是有意义的。

11.2 感知器

感知器(perceptron)是基本处理元素。它具有输入，其输入可能来自环境或者可以是其他感知器的输出。与每个输入 $x_j \in \mathfrak{R}(j=1, \cdots, d)$ 相关联的是一个连接权重(connection weight)或突触权重(synaptic weight) $w_j \in \mathfrak{R}$，而输出 y 在最简单情况下是输入的加权和(参见图 11-1)：

$$y = \sum_{j=1}^{d} w_j x_j + w_0 \tag{11.1}$$

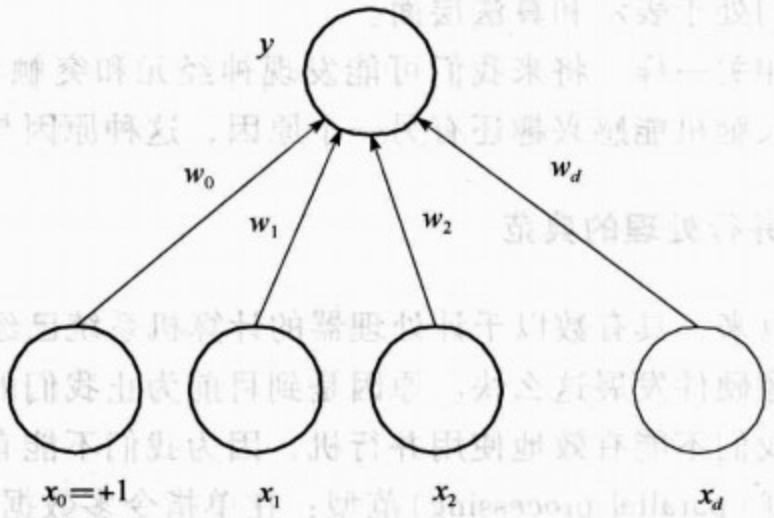


图 11-1 简单感知器，$x_j(j=1, \cdots, d)$ 是输入单元，x_0 是偏倚单元，其值总是为 1。y 是输出单元。w_j 是从 x_j 到输出的有向连接的权重

232
≀
233

其中 w_0 是截距值，它使模型更一般；通常把它作为一个来自附加的偏倚单元(bias unit) x_0 的权重，而 x_0 总是为 +1。我们可以把感知器的输出写成点积

$$y = \boldsymbol{w}^T \boldsymbol{x} \tag{11.2}$$

其中 $\boldsymbol{w} = [w_0, w_1, \cdots, w_d]^T$，$\boldsymbol{x} = [1, x_1, \cdots, x_d]^T$ 是增广向量(augmented vector)，包含偏倚权重和输入。

在检验时，给定权重 $\boldsymbol{w}$，对于输入 $\boldsymbol{x}$ 我们计算输出 y。为了实现给定的任务，我们需要学习系统的参数权重 $\boldsymbol{w}$，使得我们可以产生给定输入的正确输出。

当 $d=1$ 并且 x 通过输入单元由环境馈入时，我们有

$$y = wx + w_0$$

这是以 w 为斜率、w_0 为截距的直线方程。这样，这种具有一个输入和一个输出的感知器可以用来实现线性拟合。使用多个输入，直线变成了(超)平面，而具有多个输入的感知器可以实现多元线性拟合。给定样本，通过回归可以找出参数 w_j(见 5.8 节)。

(11.1)式定义的感知器定义了一个超平面，因此可以用来将输入空间划分成两部分：y 值为正的半个空间和 y 值为负的半个空间(参见第 10 章)。通过用它实现线性判别函数，检查输出的符号，感知器可以将两个类分开。如果我们定义 $s(\cdot)$ 为阈值函数(threshold function)

$$s(a) = \begin{cases} 1 & \text{如果 } a>0 \\ 0 & \text{否则} \end{cases} \tag{11.3}$$

则如果 $s(\boldsymbol{w}^T\boldsymbol{x})>0$ 我们可以选择 C_1，否则选择 C_2。

记住使用线性判别式假定类是线性可分的。也就是说，假定可以找到分开 $\boldsymbol{x}^t \in C_1$ 和 $\boldsymbol{x}^t \in C_2$ 的超平面 $\boldsymbol{w}^T\boldsymbol{x}=0$。如果在后一阶段我们需要后验概率(例如计算风险)，我们需要在输出使用 S 型函数

$$o = \boldsymbol{w}^T\boldsymbol{x}$$

$$y = \text{sigmoid}(o) = \frac{1}{1+\exp[-\boldsymbol{w}^T\boldsymbol{x}]} \tag{11.4}$$

234

当存在 $K>2$ 个输出时，有 K 个感知器，每个都具有权重向量 $\boldsymbol{w}_i$(参见图 11-2)

$$y_i = \sum_{j=1}^{d} w_{ij}x_j + w_{i0} = \boldsymbol{w}_i^T\boldsymbol{x}$$

$$\boldsymbol{y} = \mathbf{W}\boldsymbol{x} \tag{11.5}$$

其中 w_{ij} 是从输入 x_j 到输出 y_i 的连接权重。$\mathbf{W}$ 是 w_{ij} 的 $K\times(d+1)$ 矩阵，其行是 K 个感知器的权重向量。当用于分类时，在检验阶段，如果 $y_i = \max_k y_k$，则我们选择 C_i。

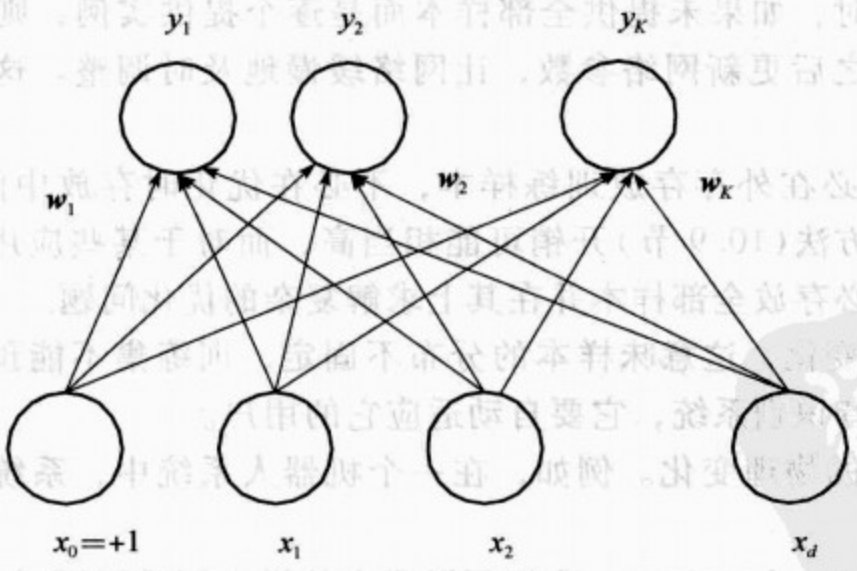


图 11-2 K 个并行的感知器。$x_j(j=0, \cdots, d)$ 是输入，$y_i(i=1, \cdots, K)$ 是输出。w_{ij}[⊖] 是从输入 x_j 到输出 y_i 的连接权重。每个输出都是输入的加权和。当用于 K 类问题时，有一个后处理，选择最大的，或者需要后验概率时选择软最大

⊖ 图中并未标出 w_{ij}，但标出了权重向量 $\boldsymbol{w}_1$，$\boldsymbol{w}_2$，$\cdots$，$\boldsymbol{w}_K$。——取自作者勘误

在神经网络中，每个感知器的值是它的输入和它的突触权重的局部函数(local function)。然而在分类中，如果我们需要后验概率(而不仅是获胜者类的编码)并使用软最大，则我们还需要其他输出值。这样，为了将其作为神经网络实现，我们可以将其看作一个两阶段过程，其中
235 第一阶段计算加权和，而第二阶段计算软最大值；但是我们仍然将其表示成单个输出单元层：

$$o_i = \boldsymbol{w}_i^T \boldsymbol{x}$$

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k} \tag{11.6}$$

回忆一下，通过定义附加的输入，例如，定义 $x_3 = x_1^2$，$x_4 = x_2^2$，$x_5 = x_1 x_2$(10.2节)，线性模型也可以用于多项式逼近。对于感知器也可以这样做(Durbin 和 Rumelhart 1989)。在11.5节，我们将看到多层感知器，那里非线性函数从数据中学习，而不是先验假定。

第10章讨论的线性判别式的任何方法都可以离线地计算 $\boldsymbol{w}_i$，$i = 1, \cdots, K$，然后插入到网络中。这包括具有公共些方差矩阵的参数方法、逻辑斯谛判别式、借助于回归的判别式和支持向量机。在某些情况下，在训练开始时我们并没有全部样本，并且随着新的实例到来，我们需要迭代地更新参数；我们将在11.3节讨论这种在线学习。

(11.5)式定义了一个从 d-维空间到 K-维空间的变换，如果 $K < d$，它也可以用于维归约。我们可以使用第6章中的任何方法(如PCA)离线地计算 $\mathbf{W}$，然后使用感知器实现变换。在这种情况下，我们有两层网络，其中第一层感知器实现线性变换，而第二层在新空间实现线性回归或分类。注意，由于两层都是线性变换，因此它们可以组合并用一层表示。在11.5节，我们将看到更有趣的情况，其中第一层实现非线性维归约。

11.3 训练感知器

感知器定义了一个超平面，而神经网络感知器只不过是实现超平面的一种方法。给定数据样本，权重可以离线地计算，并且将它们代入后，感知器就可以用来计算输出的值。

在训练神经网络时，如果未提供全部样本而是逐个提供实例，则我们通常使用在线学
236 习，并且在每个实例之后更新网络参数，让网络缓慢地及时调整。这种方法是令人感兴趣的，有如下原因：

1. 这使得我们不必在外存存放训练样本，不必在优化时存放中间结果。对于大样本，像支持向量机这样的方法(10.9节)开销可能相当高，而对于某些应用，我们可能更愿意选择较简单的方法，不必存放全部样本并在其上求解复杂的优化问题。

2. 问题可能随时变化，这意味样本的分布不固定，训练集不能预先选定。例如，我们可能正在实现一个语音识别系统，它要自动适应它的用户。

3. 可能存在系统的物理变化。例如，在一个机器人系统中，系统部件可能磨损，传感器可能失灵。

对于在线学习(online learning)，我们不需要全部样本而是需要单个实例上的误差函数。从随机初始权重开始，在每次迭代中，我们都对参数稍加调整，以最小化误差，而不忘记我们先前学到的。如果误差函数是可微的，则我们可以使用梯度下降。

例如，对于回归，单个具有标引 t 的实例 $(\boldsymbol{x}^t, r^t)$ 的误差为

$$E^t(\boldsymbol{w} \mid \boldsymbol{x}^t, r^t) = \frac{1}{2}(r^t - y^t)^2 = \frac{1}{2}[r^t - (\boldsymbol{w}^T \boldsymbol{x}^t)]^2$$

并且对于 $j=0, \cdots, d$，在线更新为

$$\Delta w_j^t = \eta(r^t - y^t)x_j^t \tag{11.7}$$

其中 η 是学习因子，随收敛逐渐减小。这称作随机梯度下降(stochastic gradient descent)。

类似地，可以对使用逻辑斯谛判别式的分类问题导出更新规则。在那里，每个模式之后进行更新，而不是把它们累加在一起，在完全扫描整个训练集之后再进行更新。对于两个类，对单个实例 $(\boldsymbol{x}^t, r^t)$，其中如果 $\boldsymbol{x}^t \in C_1$ 则 $r^t = 1$，如果 $\boldsymbol{x}^t \in C_2$ 则 $r^t = 0$，单个输出为

$$y^t = \text{sigmoid}(\boldsymbol{w}^T\boldsymbol{x}^t)$$

237

而互熵为

$$E^t(\{w_i\}_i \mid \boldsymbol{x}^t, r^t) = -r^t \log y^t + (1 - r^t)\log(1 - y^t)$$

使用梯度下降，对于 $j=0, \cdots, d$，我们得到如下更新规则：

$$\Delta w_j^t = \eta(r^t - y^t)x_j^t \tag{11.8}$$

当存在 $K>2$ 个类时，对单个实例 $(\boldsymbol{x}^t, \boldsymbol{r}^t)$，其中如果 $\boldsymbol{x}^t \in C_i$ 则 $r_i^t = 1$，否则 $r_i^t = 0$，输出为

$$y_i^t = \frac{\exp \boldsymbol{w}_i^T\boldsymbol{x}^t}{\sum_k \exp \boldsymbol{w}_k^T\boldsymbol{x}^t}$$

而互熵为

$$E^t(\{w_i\}_i \mid \boldsymbol{x}^t, r^t) = -\sum_i r_i^t \log y_i^t$$

使用梯度下降，对于 $i=1, \cdots, K$，$j=0, \cdots, d$，我们得到如下更新规则：

$$\Delta w_{ij}^t = \eta(r_i^t - y_i^t)x_j^t \tag{11.9}$$

除了不在所有的实例上求和，而是在单个实例之后更新外，这与我们在 10.7 节中看到的方程一样。算法的伪代码在图 11-3 中，它是图 10-8 算法的在线版本。

For $i=1, \cdots, K$
For $j=0, \cdots, d$
    $w_{ij} \leftarrow \text{rand}(-0.01, 0.01)$
Repeat
For 随机序下的所有 $(\boldsymbol{x}^t, r^t) \in \mathcal{X}$
For $i=1, \cdots, K$
        $o_i \leftarrow 0$
For $j=0, \cdots, d$
        $o_i \leftarrow o_i + w_{ij}x_j^t$
For $i=1, \cdots, K$
    $y_i \leftarrow \exp(o_i) / \sum_k \exp(o_k)$
For $i=1, \cdots, K$
For $j=0, \cdots, d$
    $wij \leftarrow w_{ij} + \eta(r_i^t - y_i)x_j^t$
Until 收敛

图 11-3　对于具有 $K>2$ 个类的情况，实现随机梯度下降的感知器训练算法。这是图 10-8 中给出的算法的在线版本

(11.7)和(11.9)式都具有如下形式

$$\text{更新} = \text{学习因子} \times (\text{期望输出} - \text{实际输出}) \times \text{输入} \tag{11.10}$$

让我们更深入地考察上式。首先，如果实际输出等于期望输出，则不需要更新。当进行更新时，更新随期望输出与实际输出的差增加而增加。我们还看到，如果实际输出小于期望输出，则当输入为正时更新为正，输入为负时更新为负。这具有增加实际输出和降低差别的效果。如果实际输出大于期望输出，则当输入为正时更新为负，输入为负时更新为正；这就降低了实际输出，使得它更接近于期望输出。

在做更新时，更新量还依赖于输入。如果输入接近于0，则它对实际输出的影响很小，因此其权重用一个较小的量更新。输入越大，其权重的更新也越大。

最后，更新量依赖于学习因子 η。如果它太大，则更新过分依赖当前实例；就像系统只有短期记忆。如果该因子太小，则可能需要很多次更新才收敛。在11.8.1节，我们将讨论加快收敛的方法。

11.4 学习布尔函数

在布尔函数中，输入是二元的，并且如果对应的函数值为真则输出为1，否则为0。这样，它可以看作两类分类问题。作为一个例子，考虑学习AND两个输入，输入表和期望输出显示在表11-1中。实现AND的感知器和它的二维几何表示的一个例子显示在图11-4中。判别式是

238
≀
239

$$y = s(x_1 + x_2 - 1.5)$$

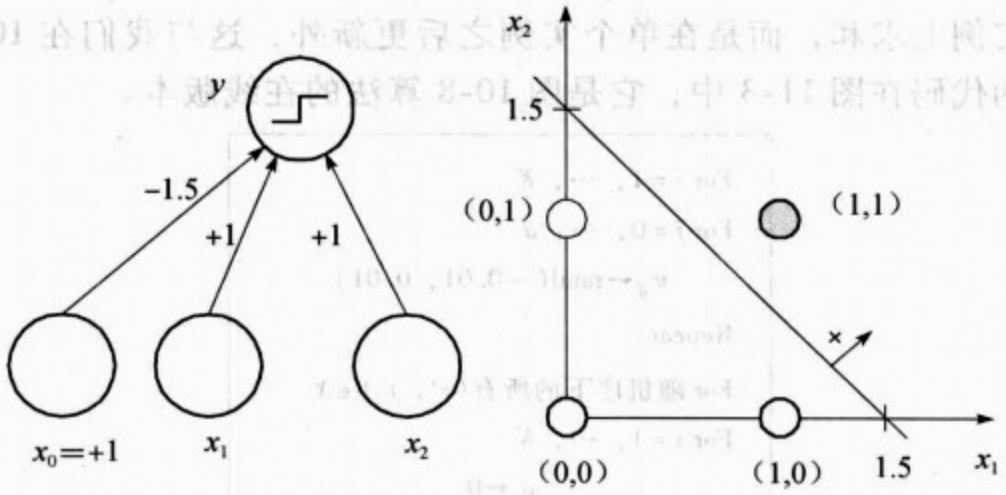


图11-4 实现AND的感知器和它的几何表示

也就是说 $\boldsymbol{x} = [1, x_1, x_2]^T$，$\boldsymbol{w} = [-1.5, 1, 1]^T$。注意，$y = s(x_1 + x_2 - 1.5)$满足表11-1中AND函数定义给定的四个约束条件。例如，对于 $x_1 = 1$，$x_2 = 0$，$y = s(-0.5) = 0$。类似地，可以证明 $y = s(x_1 + x_2 - 0.5)$实现OR。

表11-1 AND函数的输入和输出

x_1	x_2	r
0	0	0
0	1	0
1	0	0
1	1	1

尽管像 AND 和 OR 这样的布尔函数是线性可分的，并且是可以用感知器求解的，但是像 XOR 这样的函数不是。XOR 的输入和期望输出在表 11-2 中。正如我们可以从图 11-5 看到的，该问题不是线性可分的。可以证明这一点。注意不存在 w_0，w_1 和 w_2 的值满足下列不等式：

$$
\begin{aligned}
w_0 &\leqslant 0 \\
w_2 + w_0 &> 0 \\
w_1 + w_0 &> 0 \\
w_1 + w_2 + w_0 &\leqslant 0
\end{aligned}
$$

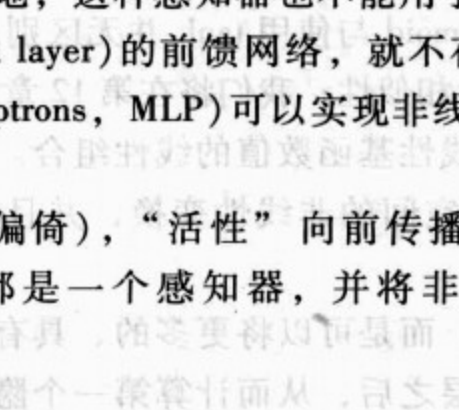


图 11-5　XOR 问题不是线性可分的。我们不能划一条直线使得空心圆在一侧，实心圆在另一侧

表 11-2　XOR 函数的输入和输出

x_1	x_2	r
0	0	0
0	1	1
1	0	1
1	1	0

我们对这一结果并不奇怪，因为(二维)直线的 VC 维为三。具有二元输入，存在四种情况，因此我们知道存在具有两个输入的问题，它们不能用直线求解；XOR 就是其中之一。

11.5　多层感知器

具有单层权重的感知器只能逼近输入的线性函数，不能解决像 XOR 这样的问题，这些问题的判别式是非线性的。类似地，这种感知器也不能用于非线性回归。对于输入和输出层之间存在中间层或*隐藏层*(hidden layer)的前馈网络，就不存在这种局限性。如果用于分类，这种*多层感知器*(multilayer perceptrons，MLP)可以实现非线性判别式，而如果用于回归，可以逼近输入的非线性函数。

输入 $\boldsymbol{x}$ 提供给输入层(包括偏倚)，“活性”向前传播，并计算隐藏单元的值 z_h(参见图 11-6)。每个隐藏单元自身都是一个感知器，并将非线性的 S 形函数作用于它的加权和：

240
≀
241

$$z_h = \text{sigmoid}(\boldsymbol{w}_h^T \boldsymbol{x}) = \frac{1}{1 + \exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}, \quad h = 1, \cdots, H \tag{11.11}$$

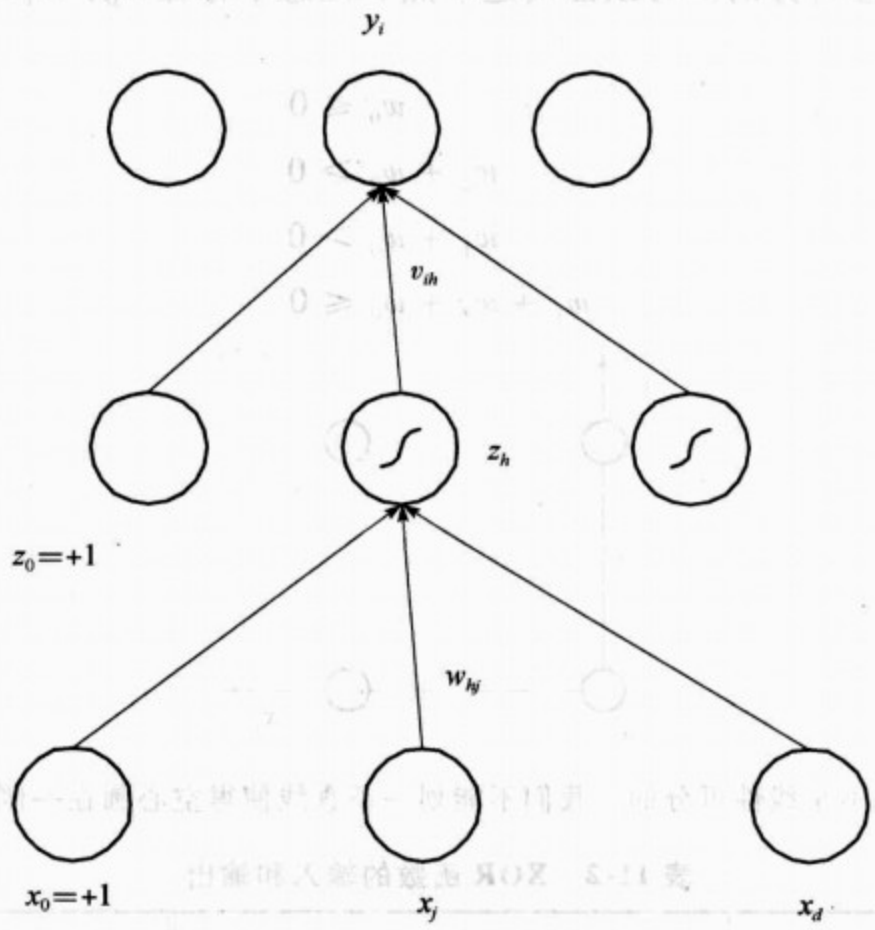


图 11-6 多层感知器结构。$x_j (j=0, \cdots, d)$是输入，$z_h (h=1, \cdots, H)$是隐藏单元，其中 H 是隐藏空间的维度。z_0 是隐藏层的偏倚。$y_i (i=1, \cdots, K)$是输出单元。w_{hj}是第一层的权重，而 v_{ih}是第二层的权重

输出 y_i 是在第二层的感知器，取隐藏单元作为它们的输入

$$y_i = \boldsymbol{v}_i^T \boldsymbol{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0} \tag{11.12}$$

其中隐藏层还有一个偏倚单元，记作 z_0，而 v_{i0}是偏倚权重。输入层单元 x_j 不计，因为计算不在那里进行，并且当有一个隐藏层时，网络是两层网络。

与通常一样，在回归问题中，在计算 y 的输出层不存在非线性。在两类判别式任务中，有一个 S 形输出单元，并且在存在 $K>2$ 个类时，有 K 个以软最大作为输出非线性的输出。

如果隐藏层单元的输出是线性的，则隐藏层就没有用：线性组合的线性组合还是一种线性组合。S 形函数是取阈值的连续、可微版本。我们需要可微性，因为我们将看到学习方程是基于梯度的。另一种可以使用的 S 形非线性基函数是双曲正切函数 tanh，它值域是 -1 到 +1，而不是 0 到 +1。在实践中，使用 sigmoid 与使用 tanh 并无区别。还有一种可能是使用高斯函数，它使用欧氏距离而不是用点积表示相似性；我们将在第 12 章讨论这种径向基函数网络。

输出是隐藏层单元计算的非线性基函数值的线性组合。可以说隐藏单元做了一个从 d-维输入空间到隐藏单元生成的 H-维空间的非线性变换，并且在这个空间中，输出层实现了一个线性函数。

我们不限于只有一个隐藏层，而是可以将更多的、具有自己的输入权重的隐藏层放置在具有 S 形隐藏单元的第一个隐藏层之后，从而计算第一个隐藏单元层的非线性函数，实现输

入的更复杂的函数。实践中，人们很少构建超过一个隐藏层的网络，因为分析多个隐藏层的网络相当复杂。但是，有时隐藏层包含的隐藏单元太多时，使用多个隐藏层可能是明智的，宁可要“长而窄”的网络，也不要“短而胖”的网络。

240 ~ 243

11.6 MLP 作为通用逼近器

我们可以将任意布尔函数表示成合取的析取，这样一个布尔表达式可以用一个具有一个隐藏层的多层感知器实现。每个合取用一个隐藏单元实现，而析取用输出单元实现。例如，

$$x_1 \text{XOR } x_2 = (x_1 \text{AND } \sim x_2) \text{OR} (\sim x_1 \text{AND } x_2)$$

前面我们已经看到如何使用感知器实现 AND 和 OR。因此，两个感知器可以平行地实现两个 AND，而另一个感知器可以将它们 OR 在一起(参见图 11-7)。我们看到，第一个隐藏层将输入从(x_1, x_2)映射到由第一层感知器定义的(z_1, z_2)空间。注意，输入(0, 0)和(1, 1)都被映射到(z_1, z_2)空间的(0, 0)，使得在第二个空间是线性可分的。

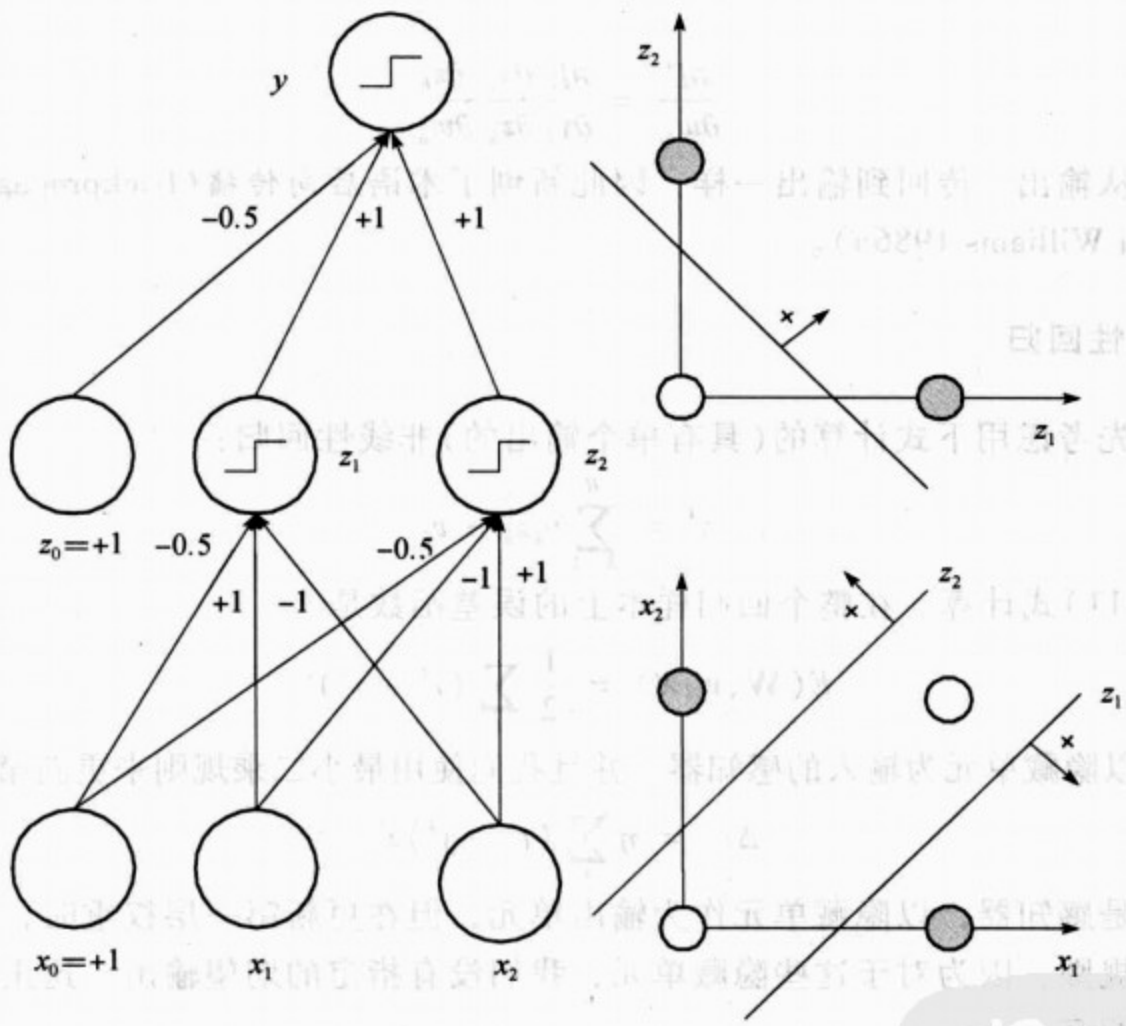


图 11-7 求解 XOR 问题的多层感知器。隐藏单元和输出单元具有阈值在 0 上的阈值活化函数

这样，在二元情况下，对于输出为 1 的每个输入组合，我们定义一个隐藏单元，它检查输入的这个特定合取。然后，输出单元实现析取。注意，这只是一个存在性证明，而这种网络可能不现实，因为当存在 d 个输入时，可能需要多达 2^d 个隐藏单元。这种结构实现了表查找而不是一般化。

我们可以将这些扩展到输入是连续值的情况，并且类似地证明具有连续输入和输出的任何函数都可以用多层感知器近似。使用两个隐藏层，通用逼近(universal approximation)的证明很容易：对于每种输入或区域，使用第一个隐藏层上的隐藏单元，该区域可以被所有边上的超平

面所界定。第二个隐藏层的单元 AND 它们，围住该区域。然后，将隐藏单元到输出单元的连接权重设置为期望的函数值。这给出函数的分段常量近似(piecewise constant approximation)；这对应忽略泰勒展开式中除常数项之外的所有项。增加隐藏单元的数量，并在输入空间中取更细的栅格，可以提高逼近期望值的精度。注意，没有给定期望的隐藏单元个数的形式上界。这种性质只是确保存在一个解；除此之外对我们并无其他帮助。业已证明，具有一个隐藏层的MLP(具有任意个数的隐藏单元)可以学习输入的任意非线性函数(Hornik、Stinchcombe 和 White 1989)。

11.7 后向传播算法

训练多层感知器与训练感知器一样；唯一的区别是现在的输出是输入的非线性函数，这要感谢隐藏单元中的非线性偏倚函数。考虑把隐藏单元作为输入，第二层是感知器，我们已经知道在给定输入 z_h 的情况下，如何更新参数 v_{ij}。对于第一层权重 w_{hj}，我们使用链规则计算梯度：

$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial z_h}\frac{\partial z_h}{\partial w_{hj}}$$

244 ≀ 245

误差就像从输出 y 传回到输出一样，因此新创了术语后向传播(backpropagation)(Rumelhart、Hinton 和 Williams 1986a)。

11.7.1 非线性回归

让我们首先考虑用下式计算的(具有单个输出的)非线性回归：

$$y^t = \sum_{h=1}^{H} v_h z_h^t + v_0 \tag{11.13}$$

其中 z_h 用(11.11)式计算。在整个回归样本上的误差函数是

$$E(\mathbf{W}, v \mid \mathcal{X}) = \frac{1}{2}\sum_t (r^t - y^t)^2 \tag{11.14}$$

第二层是以隐藏单元为输入的感知器，并且我们使用最小二乘规则来更新第二层的权重：

$$\Delta v_h = \eta \sum_t (r^t - y^t) z_h^t \tag{11.15}$$

第一层也是感知器，以隐藏单元作为输出单元，但在更新第一层权重时，我们不能直接使用最小二乘规则，因为对于这些隐藏单元，我们没有指定的期望输出。这正是链规则起作用的地方。我们有

$$\begin{aligned}
\Delta w_{hj} &= -\eta \frac{\partial E}{\partial w_{hj}} \\
&= -\eta \sum_t \frac{\partial E^t}{\partial y^t}\frac{\partial y^t}{\partial z_h^t}\frac{\partial z_h^t}{\partial w_{hj}} \\
&= -\eta \sum_t \underbrace{-(r^t - y^t)}_{\partial E^t/\partial y^t} \underbrace{v_h}_{\partial y^t/\partial z_h^t} \underbrace{z_h^t(1 - z_h^t)x_j^t}_{\partial z_h^t/\partial w_{hj}} \\
&= \eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t
\end{aligned} \tag{11.16}$$

前两项的乘积$(r^t - y^t)v_h$就像隐藏单元h的误差项。误差向后传播到隐藏单元。$(r^t - y^t)$是输出误差，按隐藏单元的“责任”加权，由其权重v_h给出。在第三项中，$z_h(1 - z_h)$是S形函数的导数，x_j^t是加权和关于权重w_{hj}的导数。注意，第一层权重的改变Δw_{hj}使用了第二层的权重v_h。因此，我们应当计算两层的改变，并更新第一层的权重，然后使用第二层权重的旧值更新第二层的权重。 246

最初，权重w_{hj}和v_h从小随机值(例如，区间$[-0.01, 0.01]$中的值)开始，使得S形函数不饱和。规范化输入使得它们都具有均值0和单位方差并且具有相同尺度也是一种好的想法，因为我们使用了单个η参数。

使用这里给定的学习方程，对于每个模式，我们计算每个参数改变的方向和改变量。在批学习(batch learning)，我们累积所有模式上的改变，并且在完全扫描了整个训练集之后做一次改变，如前面的更新方程所示。训练集中所有模式的一次完整扫描称作一个周期(epoch)。也可以在线学习，每个模式后更新权重，实现随机梯度下降。在这种情况下，应当选择较小的学习因子η，并且应当以随机次序扫描模式。因为数据集中可能有类似的模式，在线学习收敛较快，并且随机性具有增加噪声的效果，并有助于避免陷入局部极小。

为回归训练多层感知器的一个例子显示在图11-8中。随着训练继续，MLP拟合逐渐接近底层函数，并且误差降低(参见图11-9)。图11-10显示如何用隐藏单元输出的和形成MLP拟合。

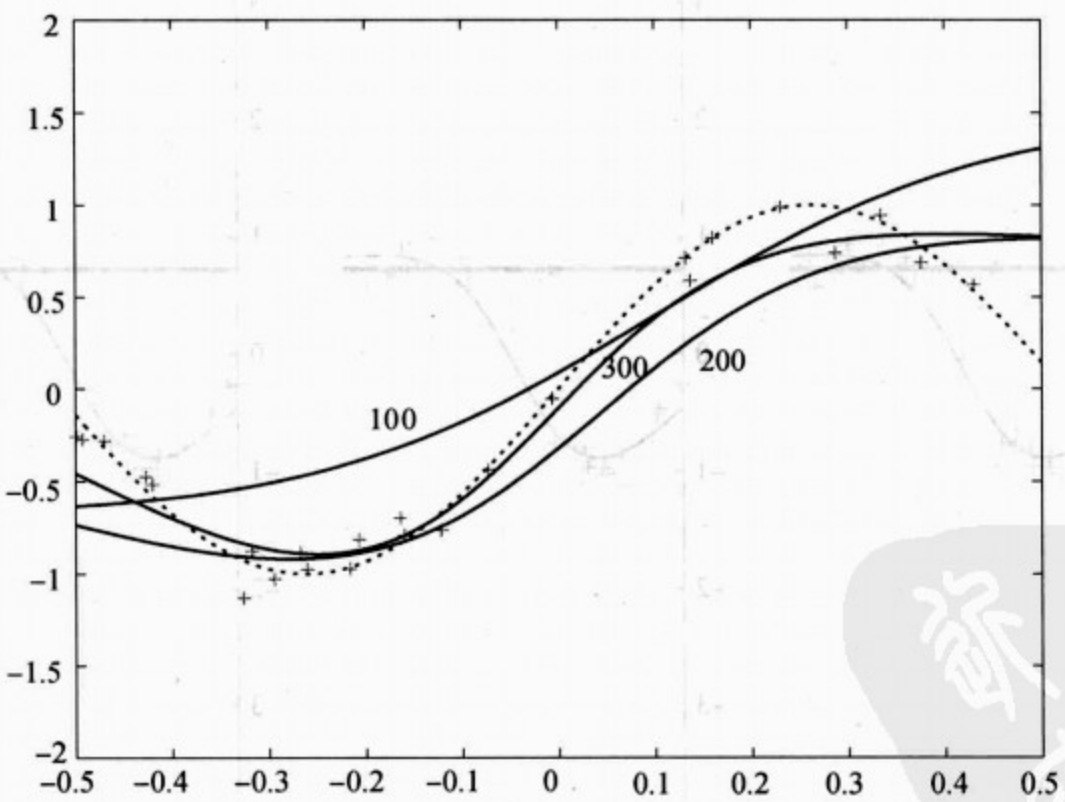


图11-8 样本训练数据显示为“+”，其中$x^t \sim U(-0.5, 0.5)$，而$y^t = f(x^t) + \mathcal{N}(0, 0.1)$。$f(x) = \sin(6x)$用虚线显示。图中绘制了100、200和300个周期后，具有两个隐藏单元的MLP的拟合演变

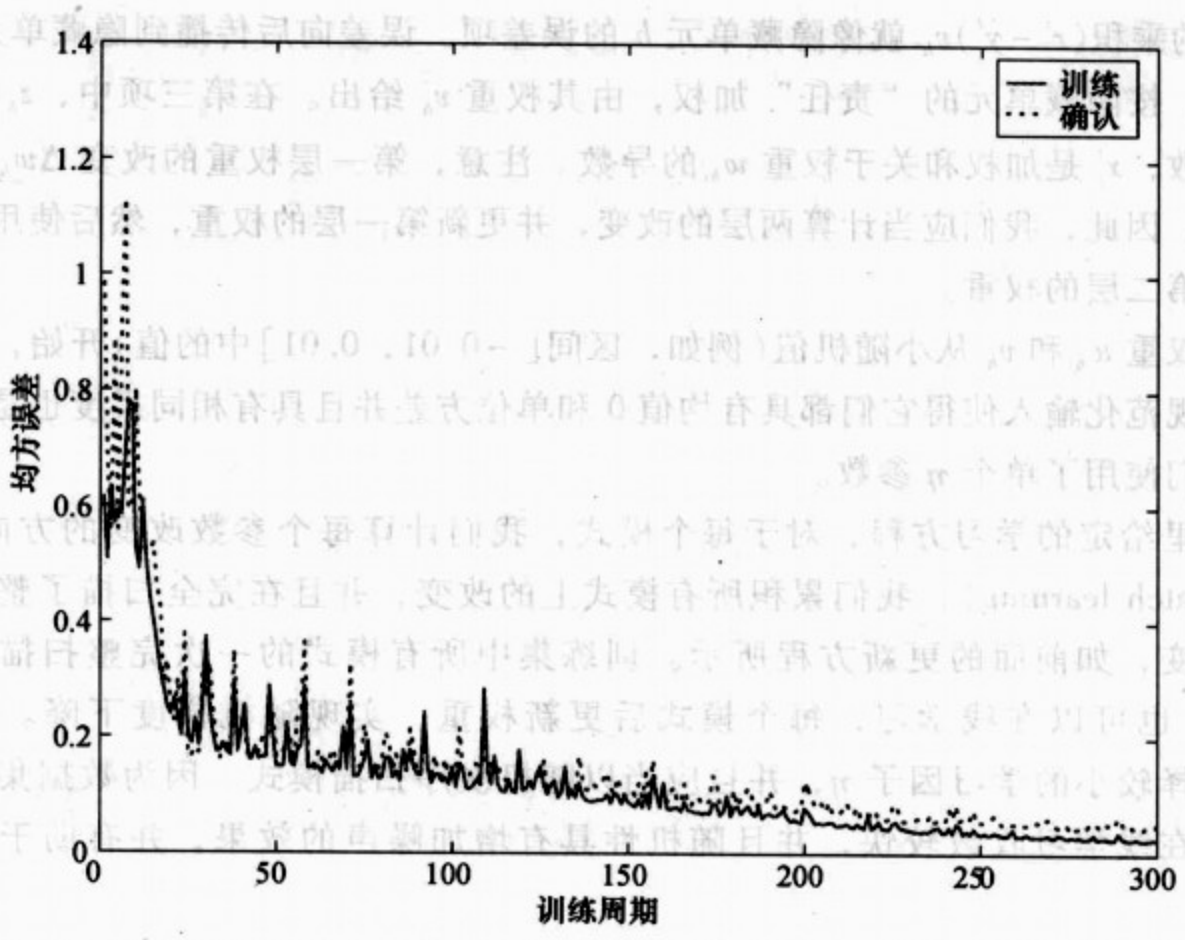


图 11-9 作为训练周期的函数，训练和确认集上的均方误差

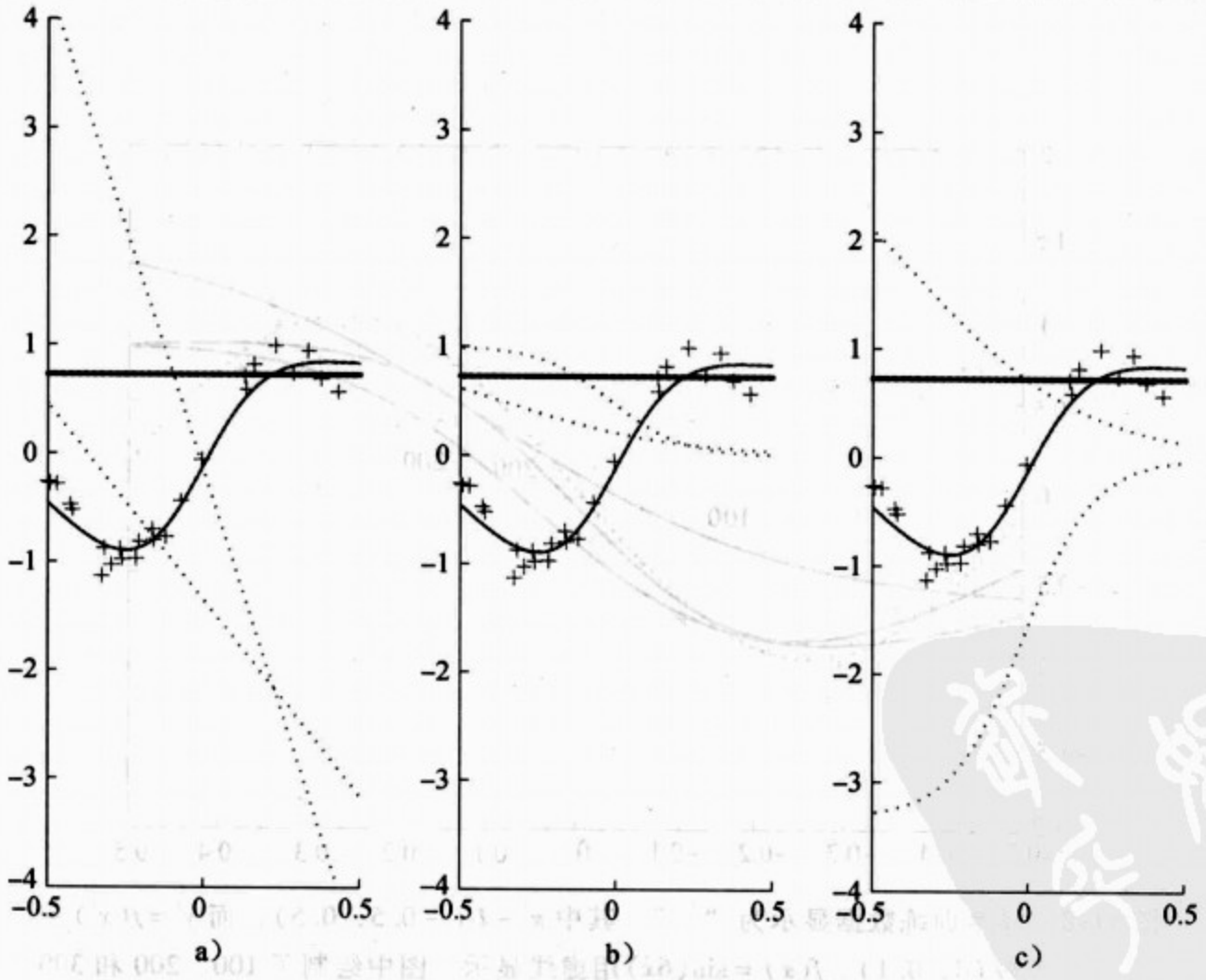


图 11-10 a)第一层上隐藏单元权重的超平面，b)隐藏单元输出，c)隐藏单元输出乘以第二层的权重。纤细线显示的两个 S 形隐藏单元，一个乘以负权重，相加时实现隆起。使用更多的隐藏单元可以得到更好的近似(参见图 11-12)

还可以有多个输出单元。在这种情况下，同时学习多个回归问题。我们有

$$y_i^t = \sum_{h=1}^{H} v_{ih} z_h^t + v_{i0} \tag{11.17}$$

而误差是

$$E(\mathbf{W},\mathbf{V}\,|\,\mathcal{X}) = \frac{1}{2}\sum_t \sum_i (r_i^t - y_i^t)^2 \tag{11.18}$$

批更新规则为

$$\Delta v_{ih} = \eta \sum_t (r_i^t - y_i^t) z_h^t \tag{11.19}$$

$$\Delta w_{hj} = \eta \sum_t \left[\sum_i (r_i^t - y_i^t) v_{ih} \right] z_h^t (1 - z_h^t) x_j^t \tag{11.20}$$

$\Sigma_i (r_i^t - y_i^t) v_{ih}$是从所有输出单元累积的误差。算法的伪代码在图 11-11 中。注意，在这种情况下，所有输出单元共享相同的隐藏单元，因此使用相同的隐藏表示。一种可供选择的方法是对每个回归问题训练一个多层感知器，每个都有自己的隐藏单元。

将所有的 v_{ih} 和 w_{hj} 初始化为 rand(-0.01, 0.01)
Repeat
For 随机次序下所有的 $(x^t, r^t) \in \mathcal{X}$
For $h = 1, \cdots, H$
$z_h \leftarrow \text{sigmoid}(\mathbf{w}_h^T x^t)$
For $i = 1, \cdots, K$
$y_i = \mathbf{v}_i^T \mathbf{z}$
For $i = 1, \cdots, K$
$\Delta \mathbf{v}_i = \eta (r_i^t - y_i^t) \mathbf{z}$
For $h = 1, \cdots, H$
$\Delta w_h = \eta \left(\sum_i (r_i^t - y_i^t) v_{ih} \right) z_h (1 - z_h) x^t$
For $i = 1, \cdots, K$
$\mathbf{v}_i \leftarrow \mathbf{v}_i + \Delta \mathbf{v}_i$
For $h = 1, \cdots, H$
$w_h \leftarrow w_h + \Delta w_h$
Until 收敛

图 11-11　为具有 K 个输出的回归训练多层感知器的后向传播算法。容易调整代码用于两类问题(设置单个 S 形输出)和 $K > 2$ 类问题(使用软最大输出)

11.7.2　两类判别式

在只有两个类时，一个输出单元就足够了：

$$y^t = \text{sigmoid}\left(\sum_{h=1}^{H} v_h z_h^t + v_0 \right) \tag{11.21}$$

它近似 $P(C_1 \,|\, x^t)$ 和 $\hat{P}(C_2 \,|\, x^t) \equiv 1 - y^t$。从 10.7 节中，我们知道在此情况下，误差函数是

$$E(\mathbf{W},\mathbf{v}\,|\,\mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t)\log(1 - y^t) \tag{11.22}$$

实现梯度下降的更新方程是

$$\Delta v_h = \eta \sum_t (r^t - y^t) z_h^t \tag{11.23}$$

$$\Delta w_{hj} = \eta \sum_t (r^t - y^t) v_h z_h^t (1 - z_h^t) x_j^t \tag{11.24}$$

与简单感知器一样，回归和分类的更新方程是相同的(这不意味它们的值相同)。

11.7.3 多类判别式

在 $K>2$ 类分类问题中，有 K 个输出

$$o_i^t = \sum_{h=1}^{H} v_{ih} z_h^t + v_{i0} \tag{11.25}$$

并且我们使用软最大指示类之间的依赖性，即它们是互斥的和穷举的：

$$y_i^t = \frac{\exp o_i^t}{\sum_k \exp o_k^t} \tag{11.26}$$

其中 y_i 近似 $P(C_i \mid \boldsymbol{x}^t)$。误差函数是

$$E(\mathbf{W}, \mathbf{V} \mid \mathcal{X}) = -\sum_t \sum_i r_i^t \log y_i^t \tag{11.27}$$

并且我们使用梯度下降得到更新方程：

$$\Delta v_{ih} = \eta \sum_t (r_i^t - y_i^t) z_h^t \tag{11.28}$$

$$\Delta w_{hj} = \eta \sum_t \left[\sum_i (r_i^t - y_i^t) v_{ih} \right] z_h^t (1 - z_h^t) x_j^t \tag{11.29}$$

Richard 和 Lippmann(1991)证明，给定一个足够复杂的网络和足够的训练数据，适当训练的多层感知器可估计后验概率。

247
~
251

11.7.4 多个隐藏层

正如我们在前面看到的，可以有多个隐藏层，每个具有自己的权重，并将 S 形函数作用于它的加权和。对于回归，假设我们有一个多层感知器，具有两个隐藏层，我们有

$$z_{1h} = \text{sigmoid}(\boldsymbol{w}_{1h}^T \boldsymbol{x}) = \text{sigmoid}\left(\sum_{j=1}^{d} w_{1hj} x_j + w_{1h0} \right), h = 1, \cdots, H_1$$

$$z_{2l} = \text{sigmoid}(\boldsymbol{w}_{2l}^T \boldsymbol{z}_1) = \text{sigmoid}\left(\sum_{h=0}^{H_1} w_{2lh} z_{1h} + w_{2l0} \right), l = 1, \cdots, H_2$$

$$y = \boldsymbol{v}^T \boldsymbol{z}_2 = \sum_{l=1}^{H_2} v_l z_{2l} + v_0$$

其中 $\boldsymbol{w}_{1h}$ 和 $\boldsymbol{w}_{2l}$ 分别是第一和第二层的权重，z_{1h} 和 z_{2h} 分别是第一和第二个隐藏层的单元，而 $\boldsymbol{v}$ 是第三层的权重。训练这种网络是类似的，唯一的区别在于，为了训练第一层的权重，我们需要后向传播更多层(习题 5)。

11.8 训练过程

11.8.1 改善收敛性

梯度下降具有多种优点。它简单。它是局部的，即权重的改变只使用前后突触单元和误

差(适合后向传播)的值。当使用在线训练时，它不需要存储训练集，并且可以自适应学习任务的变化。由于这些原因，它可以(并且已经)用硬件实现。但是，就自身而言，梯度下降收敛很慢。当学习时间很重要时，可以使用更复杂的优化方法(Battiti 1992)。Bishop (1995)详细讨论了多层感知器训练的共轭梯度的应用和二阶方法。然而，有两种频繁使用的简单技术，可以显著地改善梯度下降的性能，使得基于梯度的方法在实际应用中是可行的。 252

动量

令 w_i 为多层感知器任意层中的任意权重，包括偏倚。在每次参数更新，相继的 Δw_i^t 可能很不相同以至于可能出现摆动，减缓收敛。t 为时间指数，是批学习的周期数和在线学习的迭代次数。基本思想是在当前的改变中考虑上一次的更新，取移动平均，就好像因上次更新而存在动量(momentum)：

$$\Delta w_i^t = -\eta \frac{\partial E^t}{\partial w_i} + \alpha \Delta w_i^{t-1} \tag{11.30}$$

通常，α 在 0.5 和 1.0 之间取值。当使用在线学习时，这种方法特别有用。我们将得到平均和光滑收敛轨迹的效果。缺点是需要将过去的 Δw_i^{t-1} 存放在附加的存储中。

自适应学习率

在梯度下降中，学习因子 η 决定参数的改变量。它通常在 0.0 到 1.0 之间取值，大部分情况下小于或等于 0.2。为了更快收敛，可以让它自适应。学习进行时它保持较大，学习减慢时它也减小：

$$\Delta\eta = \begin{cases} +a & \text{如果 } E^{t+T} < E^t \\ -b\eta & \text{否则} \end{cases} \tag{11.31}$$

这样，如果训练集上的误差减小，则 η 增加一个常量；如果误差增大，则 η 减小。由于 E 可能从一个周期到另一个周期震荡，因此最好用过去几个周期的平均值作为 E^t。

11.8.2 过分训练

具有 d 个输入、H 个隐藏单元、K 个输出的多层感知器的第一层有 $H(d+1)$ 个权重，第二层有 $K(H+1)$ 个权重。MLP 的时间和空间复杂度都是 $\mathcal{O}(H \cdot (K+d))$。用 e 表示训练周期数，则训练时间复杂度为 $\mathcal{O}(e \cdot H \cdot (K+d))$。 253

在一个应用中，d 和 K 是预先确定的，H 是参数，我们用它来调整模型的复杂性。从前面的章节中我们知道，过于复杂的模型记住了训练集中的噪声，不能泛化到确认集。例如，先前我们在多项式回归中已经看到这种现象，在那里我们看到噪声或小样本的出现增加了多项式的阶，导致更糟糕的泛化。类似地，在 MLP 中，当隐藏单元数很大时，泛化精度恶化(参见图 11-12)，并且像任何统计学估计一样，对于 MLP，也存在偏倚/方差的两难选择(Geman、Bienenstock 和 Doursat 1992)。

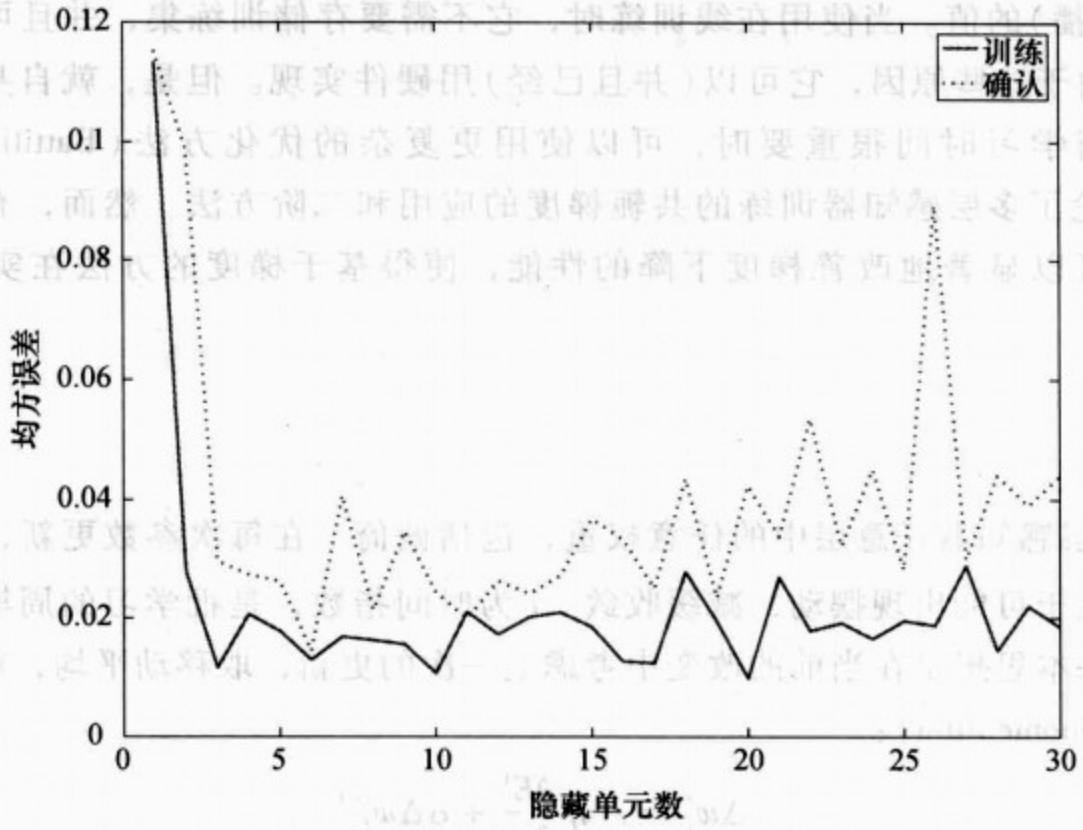


图 11-12 随着复杂度增加，训练误差固定但确认误差开始增加，网络开始过分拟合

当训练时间过长时，类似的事情也会发生：随着训练周期的增加，训练集上的误差降低，但是当超过某一点时，确认集上的误差开始增加(参见图 11-13)。回忆一下，最初所有的权重都接近于 0，因此影响很小。随着训练继续进行，大部分重要的权重开始离开 0 并发挥作用。但是，如果训练一直继续，得到训练集上越来越小的误差，几乎所有的权重都被更新，远离 0 成为有效的参数。这样，随着训练继续进行，就像将新的参数添加到系统中一样，增加了系统的复杂度，导致糟糕的泛化。学习应当在不是太晚时停止，以减轻过分训练(overtraining)问题。停止训练的最佳点和最佳隐藏单元数通过交叉确认确定，这涉及在训练期间未曾见过的确认集上测试网络的性能。

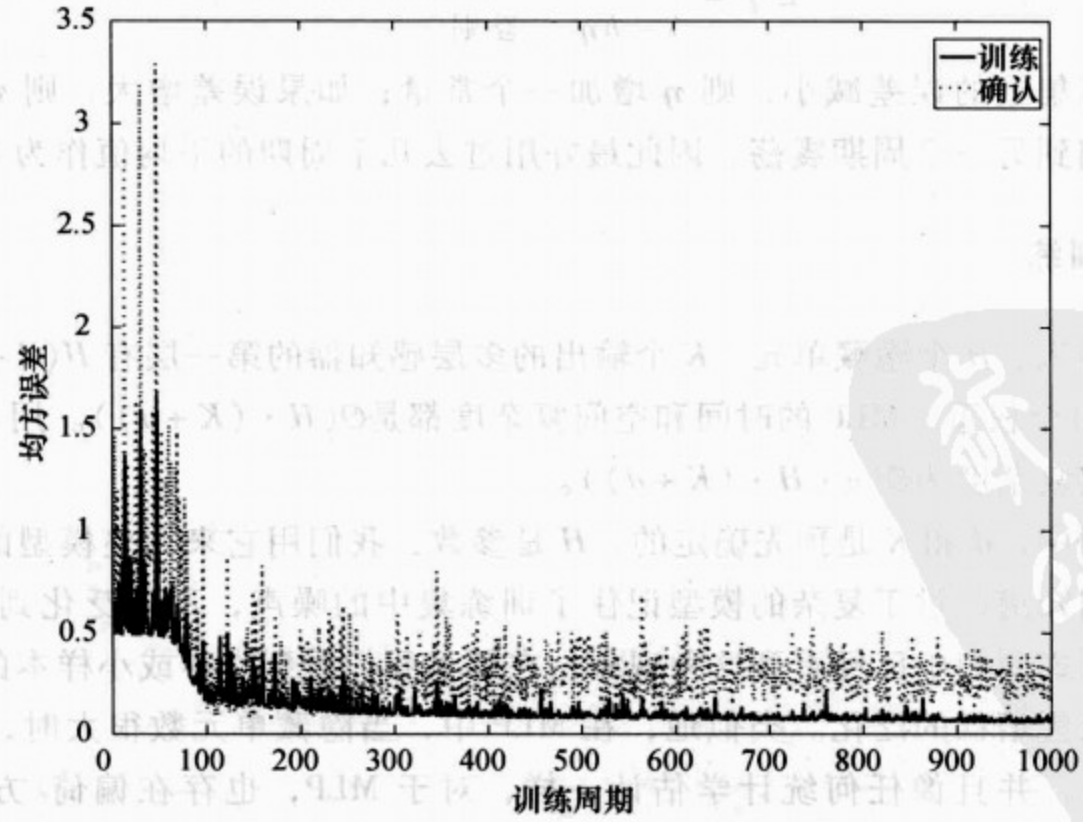


图 11-13 随着训练继续进行，确认误差开始增加，网络开始过分拟合

由于非线性性，误差函数可能具有多个极小，而梯度下降收敛于最近的极小。为了能够获得期望的误差，通常以不同的初始权重开始，对相同的网络训练多次，并且计算确认误差的平均值。

11.8.3 构造网络

在某些应用中，我们可能相信输入具有局部结构。例如，在视频中，我们知道邻近的像素是相关的，并且存在诸如边、角等局部特征。任何对象，如手写体数字，都可以定义为这些图元的组合。类似地，在语音中，存在时间局部性，并且时间上相近的输入可能组成语音基元。组合这些基元，可以定义较长的话语，如语音音素。在这种情况下，在设计 MLP 时，并不是将隐藏单元连接到所有的输入单元，因为并非所有的输入都是相关的。替换地，我们定义隐藏单元，它在输入空间上定义一个输入窗口，并且仅与输入的一个小的局部子集相连接。这样做减少了连接数，从而减少了自由参数的数目(Le Cun 等 1989)。

我们可以在相继层重复这一做法，直到输出层。每层连接下一层的少量局部单元，并且通过组合下面输入空间的较大部分，检测更复杂的特征，直到输出单元(参见图 11-14)。例如，输入可能是像素。通过观察像素，第一个隐藏层的单元可以学习检测各方向的边。然后，通过组合一些边，第二个隐藏层的单元可以学习检测边的组合(例如弧、角、线段)，并且在较高层组合它们。这些单元可以寻找半圆、矩形，或者在脸部识别应用中，寻找眼、嘴等。这是*层次锥体*(hierarchical cone)的一个例子，随着我们沿着网络向上直到我们得到类，特征越来越复杂、抽象，并且数量越来越少。

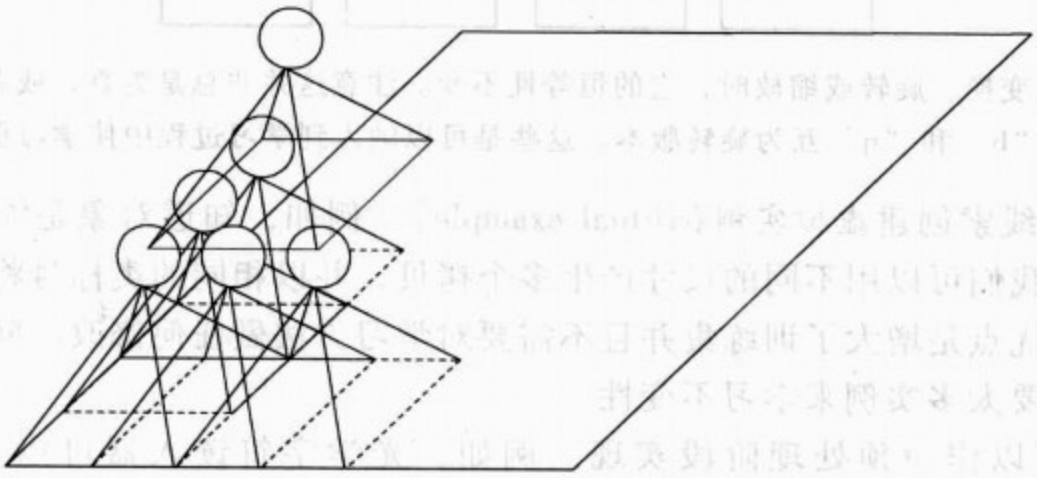


图 11-14 一个结构化的 MLP。每个单元都连接到其下单元的一个局部组群，并检测一个特定的特征(例如视频中的边、角等)。每个区域只显示了一个隐藏单元。通常，有许多隐藏单元，检测不同的局部特征

在这种情况下，我们可以通过*权重共享*(weight sharing)进一步减少参数的数目。再次以视频识别为例，我们可以看到：在我们寻找类似有向边这样的特征时，它们可能出现在输入空间的不同部分。因此，替代定义独立的隐藏单元学习输入空间不同部分的不同特征，我们可以有考察输入空间不同部分的相同隐藏单元的拷贝(参见图 11-15)。在学习期间，我们取不同的输入计算梯度，然后对它们取平均值，并做单个更新。这意味单个参数定义多个连接上的权重。此外，由于一个权重上的更新基于多个输入的梯度，因此训练集实际上就像有许多个。

254
~
256

257 ~ 258

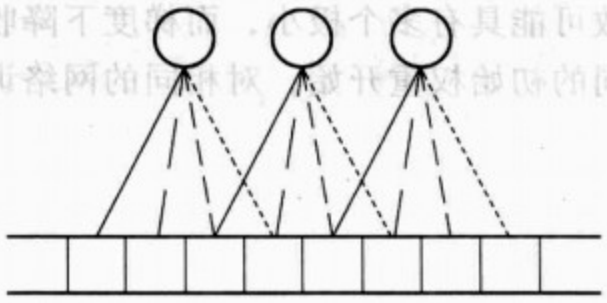


图 11-15 在权重共享中，不同的单元具有到不同输入的连接，但是共享相同的权重值（用线型表示）。只显示了一组单元；应当有多组单元，每个检测不同的特征

11.8.4 线索

局部结构的知识使得我们可以预先构造多层网络，并且使用权重共享，使得它具有较少的参数。具有全连接层的 MLP 不具有这种结构，并且更难训练。可能的话，与应用相关的任何类型的知识都应当构建到网络结构中。这些称作线索(hints)(Abu-Mostafa 1995)。它们是我们知道的目标函数的性质，独立于训练实例。

在图像识别，存在一些不变性线索：对象旋转、变换或缩放时，它的恒等性不变(参见图 11-16)。线索是辅助信息，可以用来指导学习过程，并且在训练集有限时特别有用。使用线索可以有不同的方法：

259

图 11-16 当对象变换、旋转或缩放时，它的恒等性不变。注意这并非总是为真，或者可能在某种程度为真："b" 和 "q" 互为旋转版本。这些是可以纳入到学习过程中使学习更容易的线索

1. 可以使用线索创建虚拟实例(virtual example)。例如，知道对象是缩放不变的，由给定的训练实例，我们可以用不同的尺寸产生多个拷贝，并以相同的类标号将它们添加到训练集中。这样做的优点是增大了训练集并且不需要对学习方法做任何修改。问题可能是对于学习方法，可能需要太多实例来学习不变性。

2. 不变性可以作为预处理阶段实现。例如，光学字符读入器可以有一个预处理步骤，将输入字符的图像关于尺寸和倾斜进行中心化和规范化。当可行时，这是最简单的解决方案。

3. 线索可以纳入到网络结构中。我们在 11.8.3 节看到的局部结构和权重共享就是一个例子，那里我们对小变换和旋转得到不变性。

4. 也可以通过修改误差函数纳入线索。假设我们知道从应用角度来说 x 和 x' 是相同的，其中 x' 是 x 的 "虚拟实例"。也就是说，当 $f(x)$ 是我们要逼近的函数时，$f(x) = f(x')$。让我们用 $g(x \mid \theta)$ 表示我们的逼近函数，例如 MLP，其中 θ 是它的权重。然后，对于所有这样的 (x, x')，我们定义罚函数

$$E_h = [g(x \mid \theta) - g(x' \mid \theta)]^2$$

并把它作为一个额外项加到通常的误差函数中：

$$E' = E + \lambda_h \cdot E_h$$

这是一个罚项，处罚预测不服从线索的案例，而 λ_h 是这种罚的权重(Abu-Mostafa 1995)。

另一个例子是近似线索：假设对于 $\boldsymbol{x}$，我们不知道准确的 $f(\boldsymbol{x})$ 值，但是我们知道它在区间 $[a_x, b_x]$ 中，则我们添加的罚项是：

$$E_h = \begin{cases} 0 & \text{如果 } g(x \mid \theta) \in [a_x, b_x] \\ (g(x) - a_x)^2 & \text{如果 } g(x \mid \theta) < a_x \\ (g(x) - b_x)^2 & \text{如果 } g(x \mid \theta) > b_x \end{cases}$$

这类似于支持向量机回归中使用的误差函数(10.9.4 节)，它容忍小近似误差。

还有一个例子是*正切支撑*(tangent prop)(Simard 等 1992)，其中变换与我们定义的线索相对。例如，旋转一个角度用一个函数建模。通常的误差函数被修改(添加另一个项)，使得参数可以沿着这条变换线移动而不改变误差。

11.9 调整网络规模

前面，我们看到，当网络太大，具有太多自由参数时，泛化可能不好。为了找到最佳网络规模，最常用的方法是尝试不同的结构，在训练集上训练它们，并选择对确认集泛化最好的结构。另一种方法是将*结构自适应*(structural adaptation)合并到学习算法中。有两种方法可以做这件事：

1. 在*破坏性*(destructive)方法中，我们从一个大网络开始，逐步删除不必要的单元和/或连接。

2. 在*建设性*(constructive)方法中，我们从一个小网络开始，逐步增加改善性能的单元和/或连接。

一种破坏性方法是*权衰减*(weight decay)，其基本思想是删除不必要的连接。理想情况下，为了能够确定一个单元或连接是否必要，我们需要使用它训练一次，不使用它训练一次，并检查独立的确认集上的误差之差。这种开销很大，因为这件事需要在单元/连接的所有组合上做。

假设如果一个连接的权重为 0，则没有使用它。我们给每个连接一个衰减到 0 的趋势，使得它除非为了降低误差而被明显地加强，否则就将消失。对于网络中的任意权重 w_i，我们使用更新规则：

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} - \lambda w_i \tag{11.32}$$

这等价于在具有一个附加的罚项的误差函数上做梯度下降，惩罚具有许多非零权重的网络：

$$E' = E + \frac{\lambda}{2} \sum_i w_i^2 \tag{11.33}$$

较简单的网络是较好的泛化器暗示我们通过增加一个罚项实现。注意，我们并不是说简单的网络总是比大网络好；我们是说如果我们有两个具有相同的训练误差的网络，则较简单的那个(即具有较少权重的那个)有较高的可能性更好地泛化到确认集上。

(11.32)式中第二项的效果像一个弹簧，将每个权重拉向 0。从一个接近于 0 的值开始，

除非实际误差的梯度很大并导致更新，否则由于第二项，权重将逐渐衰减为 0。λ 是参数，决定训练集上误差和由于非零参数导致的复杂性的相对重要性，因此决定衰减速度：使用大的 λ，无论训练误差多大，权重将被拉向 0；使用小的 λ，对非零权重的罚不大。使用交叉确认对 λ 进行微调。

取代从大网络开始并剪去不必要的连接或单元，我们也可以从小网络开始，必要时添加单元和相关的连接(参见图 11-17)。在*动态节点创建*(dynamic node creation)(Ash 1989)中，训练具有一个隐藏层和一个隐藏单元的网络，收敛后如果误差仍然很高，则添加一个单元。随机初始化新添加的单元的输入权重和输出权重并与先前存在的权重一起训练。先前存在的权重不再重新初始化，并从先前的值开始。

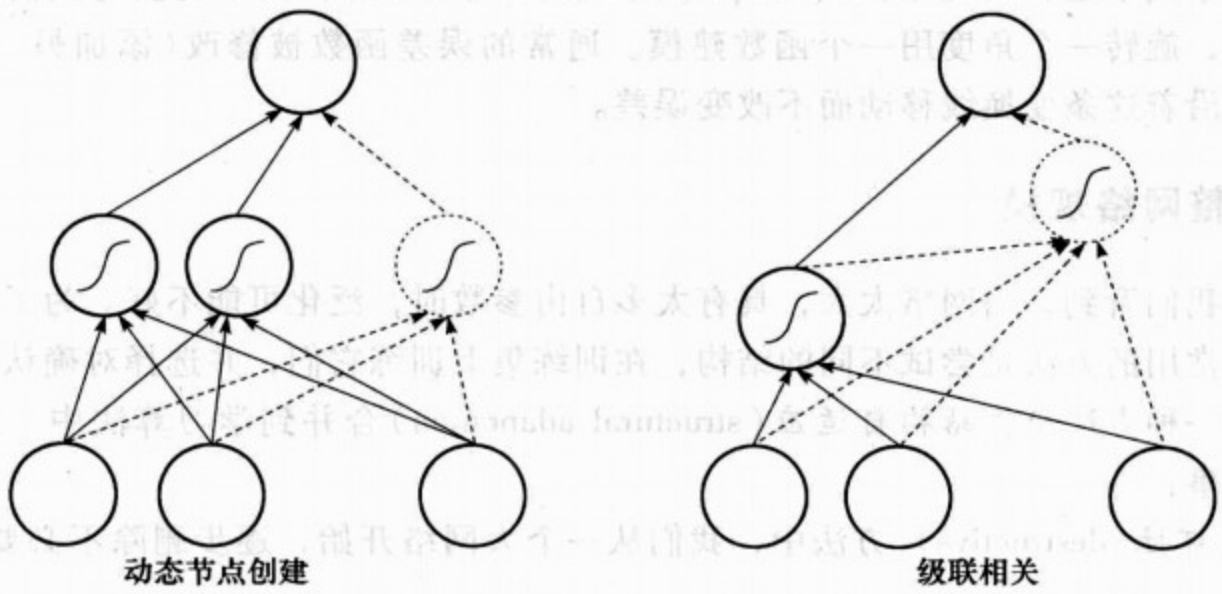


图 11-17 建设性方法的两个例子：动态节点创建向一个已存在的隐藏层添加一个单元。级联相关添加每个单元作为新的隐藏层，连接到前面的所有层。虚线表示新增加的单元/连接。为清晰起见，忽略了偏倚单元/权重

在*级联相关*(cascade correlation)中(Fahlman 和 Lebiere 1990)，每个添加的单元是另一个隐藏层中的新的隐藏单元。每个隐藏层只有一个单元连接到它前面所有隐藏单元和输入。已存在的权重被冻结，不再训练，只训练新添加的单元的输入和输出权重。

动态节点创建在已经存在的隐藏层中创建一个新的隐藏单元，而不增加新的隐藏层。级联关联总是创建具有单个单元的隐藏层。理想的建设性方法应当能够决定何时引进一个新的隐藏层，何时向已有的隐藏层添加新单元。这是一个尚待解决的研究问题。

增量算法很有趣，因为它在训练阶段不仅修改参数，而且修改模型结构。类似的思想可以用于多项式回归，其中高阶项在训练阶段自动地添加/删除，使得模型的复杂度与数据的复杂度相适应。随着计算费用逐渐降低，这种自动的模型选择将成为学习过程的一部分自动地进行，而不需要用户干预。

260
~
261

11.10 学习的贝叶斯观点

贝叶斯方法在训练神经网络时将参数(即连接权重 w_i)看作取自先验分布 $p(w_i)$ 的随机变量，并计算给定数据的后验概率

$$p(\boldsymbol{w}\mid\chi)=\frac{p(\chi\mid\boldsymbol{w})p(\boldsymbol{w})}{p(\chi)} \tag{11.34}$$

其中 $\boldsymbol{w}$ 是网络的所有权重的向量。MAP 估计 $\hat{\boldsymbol{w}}$ 为后验的众数

$$\hat{\boldsymbol{w}}_{MAP}=\arg\max_{\boldsymbol{w}}\log p(\boldsymbol{w}\mid\chi) \tag{11.35}$$

取(11.34)式的对数，我们得到

$$\log p(\boldsymbol{w}\mid\chi)=\log p(\chi\mid\boldsymbol{w})+\log p(\boldsymbol{w})+C$$

右边的第一项是对数似然，而第二项是先验概率的对数。如果权重是独立的，并且先验概率取作高斯分布$\mathcal{N}(0,\ 1/2\lambda)$

$$p(\boldsymbol{w})=\prod_i p(w_i)\quad \text{其中 } p(w_i)=c\cdot\exp\left[-\frac{w_i^2}{2(1/2\lambda)}\right] \tag{11.36}$$

则 MAP 估计最小化增广的误差函数

$$E'=E+\lambda\|\boldsymbol{w}\|^2 \tag{11.37}$$

其中 E 为通常的分类或回归误差(负的对数似然)。这个增广误差正是我们在权重衰减(11.33 式)中使用的误差函数。使用较大的 λ 意味较小的参数可变性，对它们施加更大的力量，使之接近于 0，并且更多地考虑先验而不是数据；如果 λ 较小，则允许较大的参数可变性。这种删除不必要的参数的方法在统计学中称作*岭回归*(ridge regression)。

这是使用代价函数、结合对数据的拟合和模型复杂度*正则化*(regularization)的另一个例子

$$\text{代价}=\text{数据错拟合}+\lambda\cdot\text{复杂度} \tag{11.38}$$

MacKay(1992a，b)讨论了在训练多层感知器时使用贝叶斯估计。 262

经验表明，训练后，多层感知器的大部分权重都围绕 0 正态分布，证明使用权重衰减是正确的。但是，也并非总是这种情况。Nowlan 和 Hinton(1992)提出了*软权重共享*(soft weight sharing)，其中权重取自混合高斯分布，允许它们形成多个而不是一个簇。此外，这些簇的中心可以在任何地方，而不必在 0，并且具有可以修改的方差。这将(11.36)式的先验概率改变成 $M\geqslant 2$ 个高斯混合

$$p(w_i)=\sum_{j=1}^{M}\alpha_j p_j(w_i) \tag{11.39}$$

其中 α_j 是优先权，$p_j(w_i)\sim\mathcal{N}(m_j,\ s_j^2)$是高斯分量。$M$ 由用户设置，而 α_j、m_j 和 s_j 从数据中学习。在训练阶段使用这种先验并用它的对数增广误差函数，权重收敛以降低误差，并且还自动地分组以提高对数先验。

11.11 维度归约

在多层感知器中，如果隐藏单元数小于输入数，则第一层执行维度归约。这种归约形式和隐藏单元生成的新空间依赖于 MLP 的训练目的。如果 MLP 用来分类，输出单元紧随隐藏层，则定义了新空间并且学习该映射，以降低分类误差(参见图 11-18)。

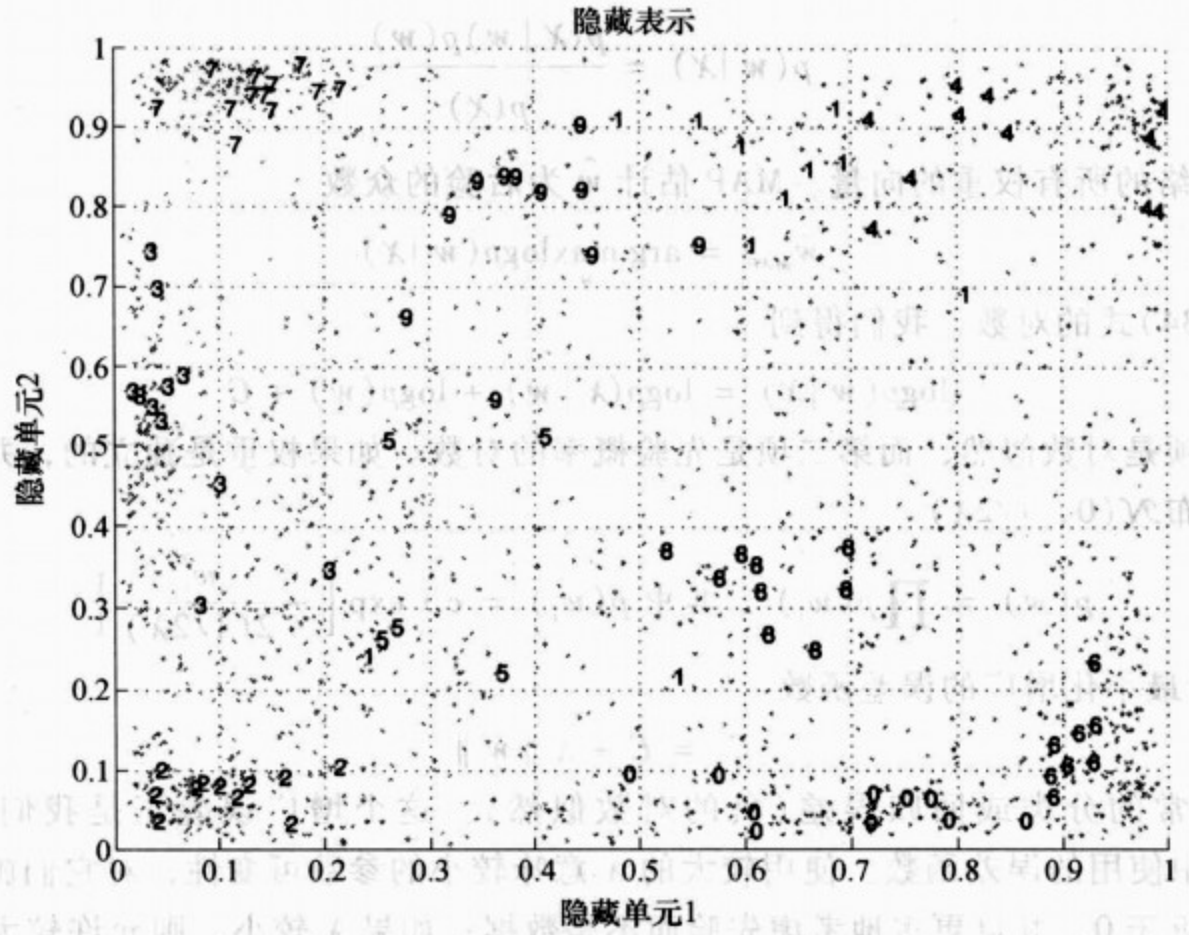


图 11-18 绘制在用于分类的训练后的 MLP 的两个隐藏单元的空间中的 Optdigits 数据。只显示了 100 个数据点的标号。该 MLP 具有 64 个输入，2 个隐藏单元和 10 个输出，具有 80% 的准确率。由于 S 形函数，隐藏单元的值在 0 和 1 之间，并且类在角落附近聚集。可以将该图与第 6 章的图比较。第 6 章的图在相同的数据集上使用其他维度归约方法绘制

通过分析权重，我们可以明白 MLP 是做什么的。我们知道当两个向量相等时点积最大。因此，我们可以认为每个隐藏单元定义了其输入权重的模板，并通过分析这些模板，我们可以从训练后的 MLP 中提取知识。如果输入是规范化的，则权重告诉我们它们的相对重要性。这样的分析并不容易，但是让我们洞察到 MLP 在做什么，并使得我们可以窥视黑箱。

一种有趣的结构是自动关联器(autoassociator)(Cottrell、Munro 和 Zipser 1987)。这是一种 MLP 结构，其中输出与输入一样多，并且所要求的输出被定义等于输入(参见图 11-19)。

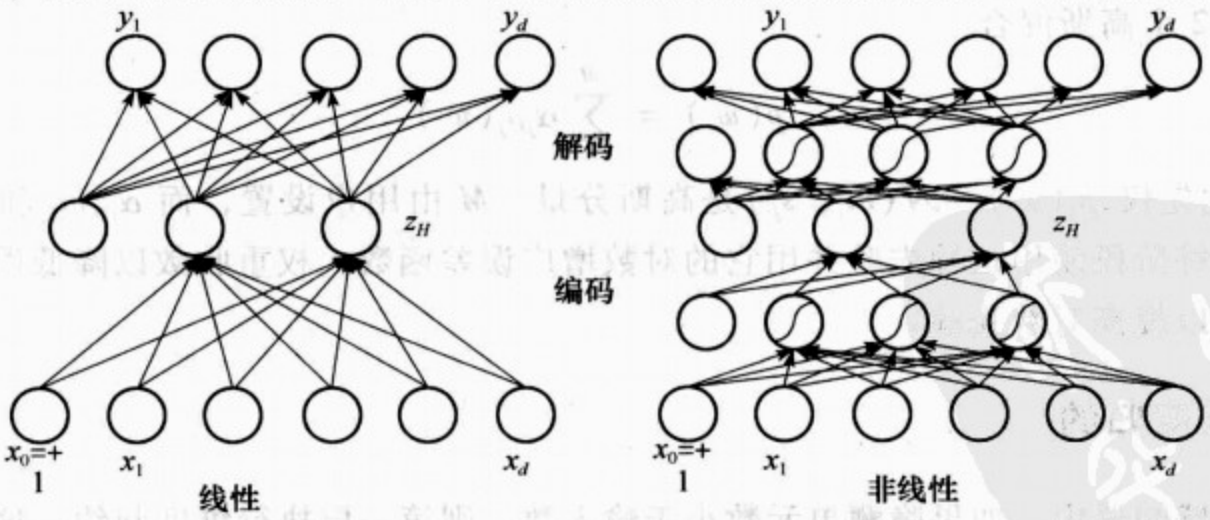


图 11-19 在自动关联器中，输出与输入一样多并且期望的输出是输入。当隐藏单元的数目小于输入的数目时，则 MLP 被训练以发现输入在隐藏层上的最佳编码，实现维度归约。左边，第一层充当编码器，而第二层充当解码器。右边，如果编码器和解码器是具有 S 形隐藏单元的多层感知器，则网络进行非线性维度归约

为了能够在输出层重新产生输入，MLP 被迫找出输入在隐藏层的最佳表示。当隐藏单元数小于输入数时，这意味着维度归约。一旦训练完成，从输入到隐藏层的第一层充当编码器，而隐藏单元的值形成编码表示。从隐藏单元到输出单元的第二层充当解码器，由原信号的编码表示重构原信号。

已经证明(Bourlard 和 Kamp 1988)，具有一个隐藏层的 MLP 实现主成分分析(6.3 节)，不同之处在于隐藏单元的权重不是使用本征值按重要性排序的本征向量，但是它与 H 个主要本征向量生成相同的空间。如果编码器和解码器不是一层，而是在隐藏单元具有 S 形非线性的多层感知器，则编码器实现非线性的维度归约。

另一种使用 MLP 进行维度归约的方法是通过多维定标(6.5 节)。Mao 和 Jain(1995)展示了如何使用 MLP 学习 Sammon 映射(Sammon mapping)。回忆(6.29)式，Sammon 应力定义为

$$E(\theta|\mathcal{X}) = \sum_{r,s}\left[\frac{\|g(\boldsymbol{x}^r|\theta) - g(\boldsymbol{x}^s|\theta)\| - \|\boldsymbol{x}^r - \boldsymbol{x}^s\|}{\|\boldsymbol{x}^r - \boldsymbol{x}^s\|}\right]^2 \tag{11.40}$$

一个具有 d 个输入，H 个隐藏单元和 $k<d$ 个输出单元的 MLP 用来实现 $g(\boldsymbol{x}|\theta)$，将 d-维输入映射到一个 k-维向量，其中 θ 对应于 MLP 的权重。给定一个数据集 $\mathcal{X}=\{\boldsymbol{x}^t\}_t$，我们可以使用梯度下降直接最小化 Sammon 应力来学习 MLP(即 $g(\boldsymbol{x}|\theta)$)，使得 k 维表示之间的距离与原空间中的距离尽可能接近。

263 ~ 265

11.12 学习时间

到目前为止，我们一直关注输入一次全部提供的情况。在某些应用中，输入是时间性数据，我们需要学习时间序列。换句话说，输出也可能随时间变化。例子有

- 序列识别(sequence recognition)。这是把给定的序列指派到多个类中的一个。语音识别是一个例子，其中输入信号序列是口语语音，而输出是词的编码。即输入随时间变化，但输出不随时间变化。
- 序列复制(sequence reproduction)。这里，在看到给定序列的一部分之后，系统将预测其余部分。时间序列产生是一个例子，那里输入是给定的，但输出是变化的。
- 时间关联(temporal association)。这是最一般的情况，其中特定的输出序列作为特定的输入序列之后的输出。输入和输出序列可能不同。这里，输入和输出都随时间变化。

11.12.1 时间延迟神经网络

识别时间序列的最简单的方法是把它转换成空间序列。然后可以利用前面讨论的任意方法进行分类。在时间延迟神经网络(time delay neural network)中(Waibel 等 1989)，前面的输入被延迟，以便与最后的输入同步，一起作为输入提交系统(参见图 11-20)。然后，使用后向传播训练权重。为了提取局部于时间的特征，我们可以使用结构化的连接层和权重共享，以便得到时间的变换不变性。这种结构的主要限制是我们滑过的序列的时间窗口大小应当预先固定。

266

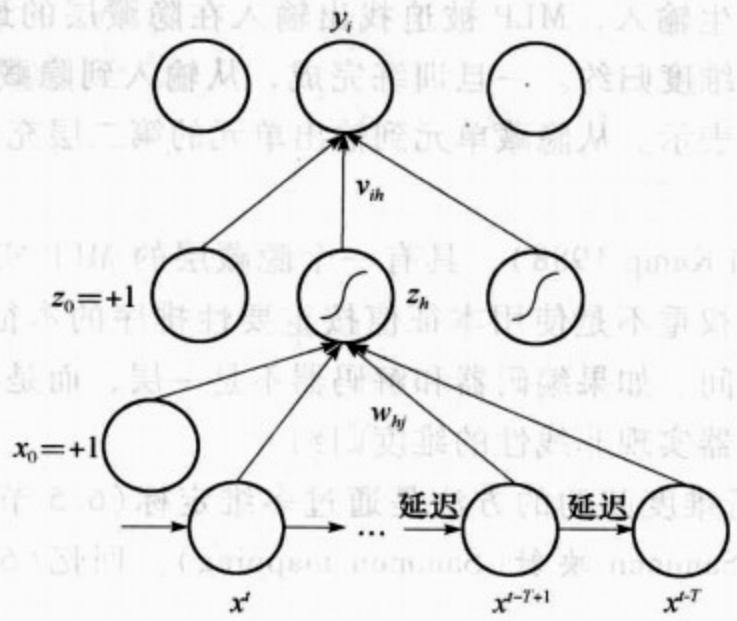


图 11-20 一个时间延迟网络。长度为 T 的时间窗口中的输入被延迟，直到我们可以将所有 T 个输入作为输入向量提供给 MLP

11.12.2 递归网络

在递归网络(recurrent network)中，除了前馈连接之外，单元具有自连接或到前面层的连接。这种递归性充当短期记忆，并使得网络记住过去发生的事。

在大部分情况下，我们使用部分递归网络，其中有限多个递归连接被添加到多层感知器中(参见图 11-21)。这结合了多层感知器的非线性逼近能力和递归的时间表达能力的优点，并且这样的网络可以用来实现三种时间关联任务中的任何一种。还可以在递归的后向连接中具有隐藏单元，这些称作上下文单元(context unit)。给定具体应用，如何选择最佳的网络结构尚无已知的正式结果。

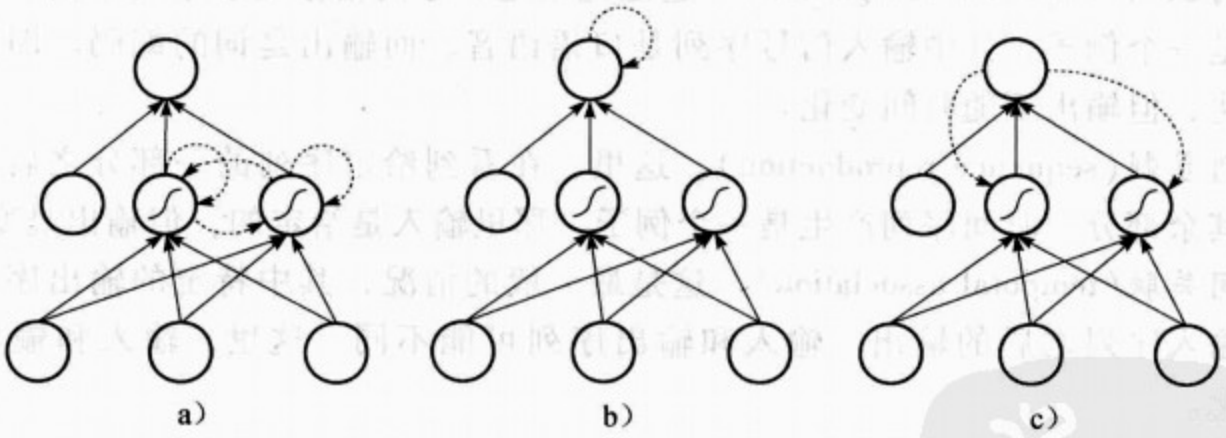


图 11-21 具有部分递归的 MLP 的例子。递归连接用虚线显示：a)隐藏层中的自连接，b)输出层中的自连接，c)从输出层到隐藏层的连接。还可以有这些情况的组合

如果序列具有较小的最大长度，则可以使用按时间展开(unfolding in time)，将任意的递归网络转换成等价的前馈网络(参见图 11-22)。为不同时间的拷贝创建单独的单元和连接。结果网络可以用后向传播训练，附加的要求是每个连接的所有拷贝应当保持相等。与权重共享一样，方法是按时间对不同权重的改变求和，并用平均值更新权重。这称作通过时间后向传播(backpropagation through time)(Rumelhart、Hinton 和 Willams 1986b)。这种方法的问题是如果序列的长度很长，则存储需求量很大。实时递归学习(real time recurrent learning)

(Williams 和 Zipser 1989)是一种训练递归网络而不展开的算法，并且具有可以用于任意长度序列的优点。

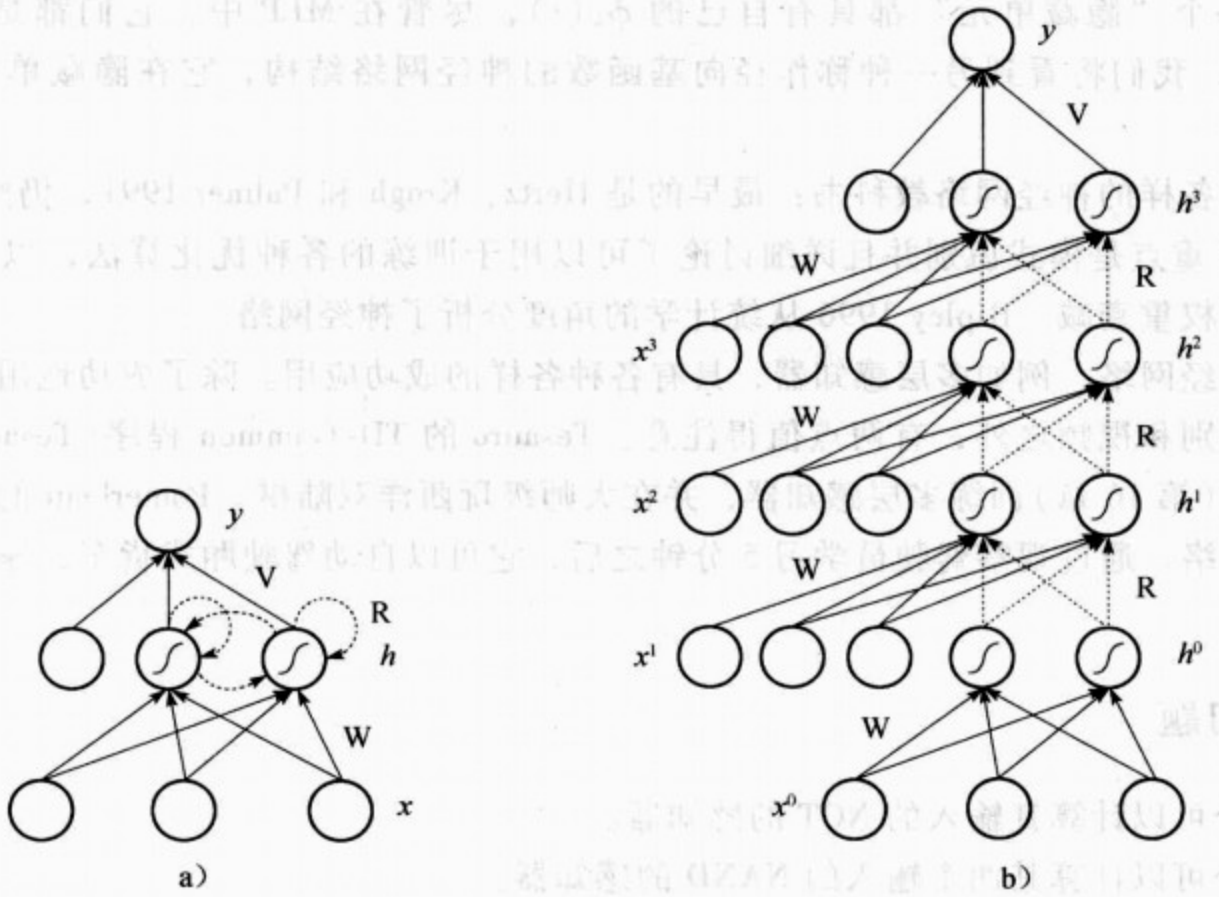


图 11-22 通过时间后向传播：a)递归网络，b)4 步中行为相同的、等价的展开网络

11.13 注释

人工神经网络的研究历史与数字计算机一样长。McCulloch 和 Pitts(1943)提出了人工神经网络的第一个数学模型。Rosenblatt(1962)提出了感知器模型和学习算法。Minsky 和 Papert(1969)指出了单层感知器的局限性(例如 XOR 问题)，并且由于那时还没有训练具有隐藏层的多层感知器的算法，因此除了少数地方之外，人工神经网络的工作几乎停止了。Hopfield(1982)的文章带来了神经网络的复兴。随后出现了并行分布处理(PDP)研究小组编写的两卷并行处理书(Rumelhart 和 McClelland 1986)。看起来，后向传播几乎同时在多个地方被发明，而单层感知器的局限性也不复存在。

从 20 世纪 80 年代中期开始，出现了关于人工神经网络模型的大量研究，来自各个学科：物理学、统计学、心理学、认知科学、神经系统科学、语言学，更不必说计算机科学、电子学和自适应控制了。或许，人工神经网络研究的最重要贡献是这种沟通不同学科，特别是沟通统计学与工程的协同。感谢这种协同，使机器学习领域现在得以确立。

现在，该领域更加成熟，目标被更适当、更好地确定。对后向传播的批评之一是，从生物学角度讲，它几乎是不可能的！尽管术语“神经网络”仍然被广泛使用，但是通常把神经网络模型理解为(例如多层感知器)非参数估计方法，并且分析它的最佳方法是使用统计学方法。

267
~
269

例如，一种类似于多层感知器的统计学方法是投影追踪(projection pursuit)(Friedman 和 Stuetzle 1981)，表示为

$$y = \sum_{h=1}^{H} \phi_h(\boldsymbol{w}_h^T \boldsymbol{x})$$

不同的是每个“隐藏单元”都具有自己的 $\phi_h(\cdot)$，尽管在 MLP 中，它们都是 S 型函数。在第 12 章，我们将看到另一种称作径向基函数的神经网络结构，它在隐藏单元使用高斯函数。

有各种各样的神经网络教科书：最早的是 Hertz、Krogh 和 Palmer 1991，仍然可以阅读。Bishop 1995 重点是模式识别并且详细讨论了可以用于训练的各种优化算法，以及贝叶斯方法，推广了权重衰减。Ripley 1996 从统计学的角度分析了神经网络。

人工神经网络，例如多层感知器，具有各种各样的成功应用。除了成功地用于自适应控制、语音识别和视频之外，有两点值得注意：Tesauro 的 TD-Gammon 程序(Tesauro 1994)使用增强学习(第 16 章)训练多层感知器，并在大师级玩西洋双陆棋。Pomerleau 的 ALVINN 是一个神经网络，通过观察驾驶员学习 5 分钟之后，它可以自动驾驶厢式货车，速度达每小时 20 英里。

11.14 习题

1. 给出一个可以计算其输入的 NOT 的感知器。
2. 给出一个可以计算其两个输入的 NAND 的感知器。
3. 给出一个可以计算其三个输入的奇偶性的感知器。
4. 当隐藏单元使用 tanh 函数而不是使用 S 形函数时，推导出更新方程。使用事实 $\tanh' = (1 - \tanh^2)$。
5. 为具有两个隐藏层的 MLP 推导更新方程。
6. 奇偶性是循环移动不变的。例如，“0101”和“1010”具有相同的奇偶性。使用这个提
270 示，提出一个学习奇偶函数的多层感知器。
7. 在级联相关中，冻结前面已经存在的权重有何优点？
8. 为实现最小化 Sammon 应力(11.40)式的 Sammon 映射的 MLP 推导更新方程。

11.15 参考文献

Abu-Mostafa, Y. 1995. “Hints.” *Neural Computation* 7: 639 – 671.

Ash, T. 1989. “Dynamic Node Creation in Backpropagation Networks.” *Connection Science* 1: 365 – 375.

Battiti, R. 1992. “First-and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method.” *Neural Computation* 4: 141 – 166.

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.

Bourlard, H., and Y. Kamp. 1988. “Auto-Association by Multilayer Perceptrons and Singular Value Decomposition.” *Biological Cybernetics* 59: 291 – 294.

Cottrell, G. W., P. Munro, and D. Zipser. 1987. “Learning Internal Representations from Gray-Scale Images: An Example of Extensional Programming.” In *Ninth Annual Conference of the Cognitive Science Society*, 462 – 473. Hillsdale, NJ: Erlbaum.

Durbin, R., and D. E. Rumelhart. 1989. “Product Units: A Computationally Powerful and Biologically Plausible

Extension to Backpropagation Networks." *Neural Computation* 1: 133 – 142.

Fahlman, S. E., and C. Lebiere. 1990. "The Cascade Correlation Architecture." In *Advances in Neural Information Processing Systems 2*, ed. D. S. Touretzky, 524 – 532. San Francisco: Morgan Kaufmann.

Friedman, J. H., and W. Stuetzle. 1981. "Projection Pursuit Regression." *Journal of the American Statistical Association* 76: 817 – 823.

Geman, S., E. Bienenstock, and R. Doursat. 1992. "Neural Networks and the Bias/Variance Dilemma." *Neural Computation* 4: 1 – 58.

Hertz, J., A. Krogh, and R. G. Palmer. 1991. *Introduction to the Theory of Neural Computation*. Reading, MA: Addison Wesley.

Hopfield, J. J. 1982. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." *Proceedings of the National Academy of Sciences USA* 79: 2554 – 2558.

Hornik, K., M. Stinchcombe, and H. White. 1989. "Multilayer Feedforward Networks Are Universal Approximators." *Neural Networks* 2: 359 – 366. 271

Le Cun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. "Backpropagation Applied to Handwritten Zipcode Recognition." *Neural Computation* 1: 541 – 551.

MacKay, D. J. C. 1992a. "Bayesian Interpolation." *Neural Computation* 4: 415 – 447.

MacKay, D. J. C. 1992b. "A Practical Bayesian Framework for Backpropagation Networks." *Neural Computation* 4: 448 – 472.

Mao, J., and A. K. Jain. 1995. "Artificial Neural Networks for Feature Extraction and Multivariate Data Projection." *IEEE Transactions on Neural Networks* 6: 296 – 317.

Marr, D. 1982. *Vision*. New York: Freeman.

McCulloch, W. S., and W. Pitts. 1943. "A Logical Calculus of the Ideas Immenent in Nervous Activity." *Bulletin of Mathematical Biophysics* 5: 115 – 133.

Minsky, M. L., and S. A. Papert. 1969. *Perceptrons*. Cambridge, MA: The MIT Press. (Expanded ed. 1990.)

Nowlan, S. J., and G. E. Hinton. 1992. "Simplifying Neural Networks by Soft Weight Sharing." *Neural Computation* 4: 473 – 493.

Pomerleau, D. A. 1991. "Efficient Training of Artificial Neural Networks for Autonomous Navigation." *Neural Computation* 3: 88 – 97.

Posner, M. I., ed. 1989. *Foundations of Cognitive Science*. Cambridge, MA: The MIT Press.

Richard, M. D., and R. P. Lippmann. 1991. "Neural Network Classifiers Estimate Bayesian *a Posteriori* Probabilities." *Neural Computation* 3: 461 – 483.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. Cambridge, UK: Cambridge University Press.

Rosenblatt, F. 1962. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. New York: Spartan.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986a. "Learning Representations by Backpropagating Errors." *Nature* 323: 533 – 536.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986b. "Learning Internal Representations by Error Propagation." In *Parallel Distributed Processing*, ed. D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, 318 – 362. Cambridge, MA: The MIT Press.

Rumelhart, D. E., J. L. McClelland, and the PDP Research Group, eds. 1986. *Parallel Distributed Processing*. Cambridge, MA: The MIT Press.

Simard, P., B. Victorri, Y, Le Cun, and J. Denker. 1992. "Tangent Prop: A Formalism for Specifying Selected Invariances in an Adaptive Network." In *Advances in Neural Information Processing Systems* 4, ed. J. E. Moody, 272 S. J. Hanson, R. P. Lippman, 895 – 903. San Francisco: Morgan Kaufmann.

Tesauro, G. 1994. "TD-Gammon, A Self-Teaching Backgammon Program, Achieves Master-Level Play." *Neural Computation* 6: 215 – 219.

Waibel, A., T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. 1989. "Phoneme Recognition Using Time-Delay Neural Networks." *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37: 328 – 339.

Williams, R. J., and D. Zipser. 1989. "A Learning Algorithm for Continually Running Fully Recurrent Neural Net- 273 works." *Neural Computation* 1: 270 – 280.

第12章 局部模型

我们继续讨论多层神经网络，考察第一层包含局部接受单元的模型；这些局部接受单元响应输入空间局部区域中的实例。上面第二层对这些局部区域学习回归或分类函数。我们讨论找出重要局部区域，以及这些区域中的模型的学习方法。

12.1 引言

进行函数逼近的一种方法是将输入空间划分成局部小片，并且在每个局部小片中分别学习拟合。在第7章，我们讨论了聚类的统计学方法，使得我们能够对输入实例分组并对输入分布建模。竞争方法是用于在线聚类的神经网络方法。本章，我们讨论 k-均值的在线版本以及两种神经网络扩展：自适应共鸣理论(ART)和自组织影射(SOM)。

然后，我们讨论一旦输入局部化，如何实现监督学习。如果局部小片上的拟合是常量，则该技术称作径向基函数(RBF)网络；如果拟合是输入的线性函数，则称作混合专家技术(MoE)。我们讨论回归和分类，并与第11章讨论的MLP方法进行比较。

12.2 竞争学习

在第7章，我们使用半参数高斯混合密度，假定输入来自 k 个高斯源中的一个。本节，我们做相同的假设，数据中存在 k 个分组(或簇)，但是我们的方法不是概率方法，因为我们不将参数模型强加在数据源上。另一个区别是我们提出的学习方法是在线的：在训练阶段我们并没有全部样本；我们逐个接收实例并更新模型参数。使用术语竞争学习(competitive learning)是因为这些分组，更确切地说，代表这些分组的单元为成为代表实例而相互竞争。这种方法也称胜者全取(winner-take-all)；就像一个分组获胜并得到更新，而其他分组则完全不更新一样。

与第7章讨论的方法相反，这些方法本身可以用于在线聚类。在线方法具有通常的优点：(1)不需要附加的存储保存整个训练集；(2)每步更新简单、易于实现(例如用硬件实现)；(3) 输入的分布可以随时间而改变，并且模型可以自动地适应这些改变。如果我们使用批处理算法，我们将需要收集新样本，并且从头开始在整个样本上运行批处理方法。

从12.3节开始，我们还将讨论这种方法如何后跟一种监督方法，来学习回归和分类问题。这将是两阶段系统，可以用两层网络实现，其中第一阶段(层)对输入密度建模并找到相应的局部模型，而第二阶段是产生最终输出的局部模型。

12.2.1 在线 k-均值

在(7.3)式，我们定义重构误差为

$$E(\{\boldsymbol{m}_i\}_{i=1}^{k} | \mathcal{X}) = \frac{1}{2}\sum_t \sum_i b_i^t \| \boldsymbol{x}^t - \boldsymbol{m}_i \|^2 \tag{12.1}$$

其中

276

$$b_i^t = \begin{cases} 1 & \text{如果 } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_l \|\boldsymbol{x}^t - \boldsymbol{m}_l\| \\ 0 & \text{否则} \end{cases} \tag{12.2}$$

$\mathcal{X} = \{\boldsymbol{x}^t\}_t$ 是样本，而 $\boldsymbol{m}_i (i = 1, \cdots, k)$ 是簇中心。如果 $\boldsymbol{m}_i$ 是 $\boldsymbol{x}^t$ 的欧氏距离最接近的中心，则 b_i^t 为 1。好像是所有的 $\boldsymbol{m}_l (l = 1, \cdots, k)$ 竞争，而 $\boldsymbol{m}_i$ 赢得竞争，因为它是最近的。

k-均值的批处理算法按下式更新中心

$$\boldsymbol{m}_i = \frac{\sum_t b_i^t \boldsymbol{x}^t}{\sum_t b_i^t} \tag{12.3}$$

一旦使用(12.2)式选取获胜者，它将最小化(12.1)式。正如我们先前看到的，计算 b_i^t 和更新 $\boldsymbol{m}_i$ 的两个步骤迭代，直到收敛。

通过进行随机梯度下降、逐个考虑实例、并在每一步进行少许更新而不忘记先前的更新，我们可以得到在线 k-均值(online k-means)。对于单个实例，重构误差为

$$E^t(\{\boldsymbol{m}_i\}_{i=1}^k \mid \boldsymbol{x}^t) = \frac{1}{2}\sum_i b_i^t \|\boldsymbol{x}^t - \boldsymbol{m}_i\|^2 = \frac{1}{2}\sum_i \sum_{j=1}^d b_i^t (x_j^t - m_{ij})^2 \tag{12.4}$$

其中 b_i^t 的定义同(12.2)式。对上式使用梯度下降，我们得到每个实例 $\boldsymbol{x}^t$ 的更新规则：

$$\Delta m_{ij} = -\eta \frac{\partial E^t}{\partial m_{ij}} = \eta b_i^t (x_j^t - m_{ij}) \tag{12.5}$$

这把最近的中心(其 $b_i^t = 1$)向输入移动一个因子 η。其他中心的 $b_i^t (l \neq i)$ 等于 0，并且不更新(参见图 12-1)。批处理过程也可以通过将 (12.5)式在所有的 t 上求和定义。与任何梯度下降过程一样，也可以添加一个动量项。为了收敛，η 逐渐减少为 0。但是，这意味稳定性与可塑性的两难选择(stability-plasticity dilemma)：如果 η 向 0 递减，则网络变得稳定，但是因为更新变得太小，我们失去了对随时出现的新模式的适应性。如果我们一直保持 η 较大，则 $\boldsymbol{m}_i$ 可能震荡。

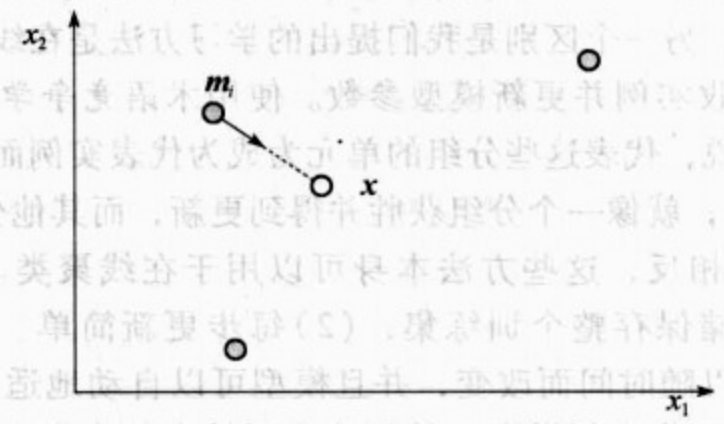


图 12-1 阴影圆是中心，空心圆是输入实例。k-均值算法的在线版本沿方向$(\boldsymbol{x} - \boldsymbol{m}_i)$将最近的中心移动一个因子 η

在线 k-均值的伪代码在图 12-2 中。这是图 7-3 的批处理算法的在线版本。

初始化 $\boldsymbol{m}_i (i = 1, \cdots, k)$，例如为 k 个随机的 $\boldsymbol{x}^t$
Repeat
  For 随机次序下所有的 $\boldsymbol{x}^t \in \mathcal{X}$
    $i \leftarrow \arg\min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\|$
    $\boldsymbol{m}_i \leftarrow \boldsymbol{m}_i + \eta(\boldsymbol{x}^t - \boldsymbol{m}_j)$
Until $\boldsymbol{m}_i$ 收敛

图 12-2 在线 k-均值算法。批处理版本在图 7-3 中

竞争网络可以用单层递归网络实现，如图 12-3 所示。输入层包含输入向量 $\boldsymbol{x}$；注意没有偏倚单元。输出单元的值是 b_i，并且它们是感知器：

$$b_i = \boldsymbol{m}_i^T \boldsymbol{x} \tag{12.6}$$

然后，我们需要选择最大的 b_i，并将它设置为 1，而设置其他 $b_l (l \neq i)$ 为 0。如果我们想用纯粹的神经系统方法做所有的事，即使用并发操作处理单元网络，则最大值的选择可以用*横向抑制*(lateral inhibition)实现。如图 12-3 所示，每个单元有一个到自身的兴奋的递归连接(即具有正权重)和到其他输出单元的抑制的递归连接(即具有负权重)。使用适当的非线性激励函数和正的及负的递归权重值，这样的网络在某些迭代后收敛于一种状态，其中最大值变成 1，而其余的变成 0(Grossberg 1980，Feldman 和 Ballard 1982)。

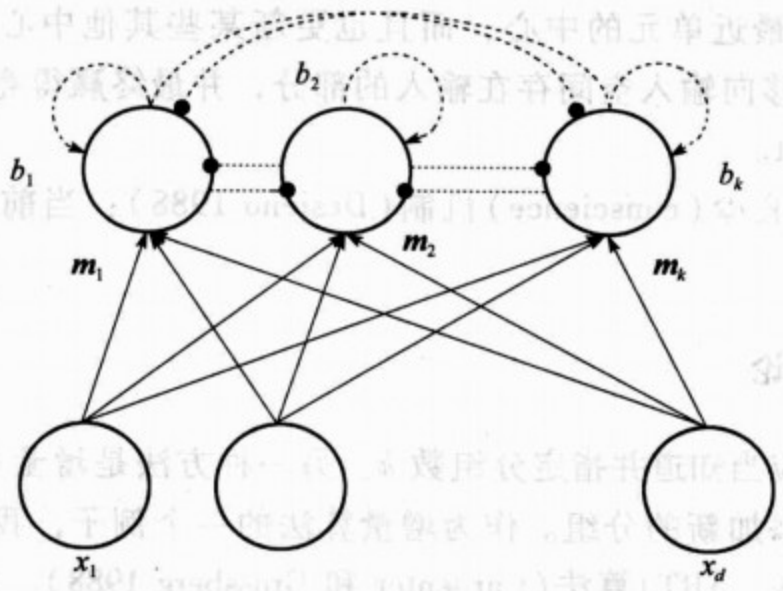


图 12-3 胜者全取竞争神经网络，它是在输出层有递归连接的 k 个感知器的网络。虚线是递归连接，其中带箭头的是兴奋的，而带圆点的是抑制的。输出层的每个单元加强它的值，并试图超过其他输出。在这些递归权重的适当赋值下，最大的抑制了其他所有的。这具有网络效应：其 $\boldsymbol{m}_i$ 最接近 $\boldsymbol{x}$ 的一个单元以其 b_i 等于 1 告终，而其他所有的 $b_l (l \neq i)$ 为 0

(12.6)式中使用的点积是一种相似性度量，并且我们在 5.5 节(5.26)式中看到，如果 $\boldsymbol{m}_i$ 具有相同的范数，则具有最小欧氏距离 $\| \boldsymbol{m}_i - \boldsymbol{x} \|$ 的单元与具有最大点积 $\boldsymbol{m}_i^T \boldsymbol{x}$ 的单元相同。

这里和后面，当我们讨论其他竞争方法时，我们使用欧氏距离，但是我们应当记住，使用欧氏距离意味所有输入属性具有相同的方差并且它们是不相关的。如果不是这种情况，则应当反映在距离度量中(即使用 Mahalanobis 距离)，或者在使用欧氏距离前，在预处理阶段做适当的规范化(例如用 PCA)。

我们可以将(12.5)式改写为

$$\Delta m_{ij}^t = \eta b_i^t x_j^t - \eta b_i^t m_{ij} \tag{12.7}$$

让我们回想一下，m_{ij}是从 x_j 到 b_i 的连接的权重。正如我们在前一项所看到的，更新

$$\Delta m_{ij}^t = \eta b_i^t x_j^t \tag{12.8}$$

是 Hebbian 学习(Hebbian learning)，它将更新定义为前突触与后突触单元值的乘积。它是作为神经可塑性模型提出的：一个突触变得更重要，如果该连接的前后单元都同时冲动，表明它们是相关的。然而，仅用 Hebbian 学习，权重的增加无界($x_j^t \geqslant 0$)，并且我们需要第二种力量，来减少未更新的权重。一种可能的方法是显式地规范化权重，使得 $\| \boldsymbol{m}_i \| = 1$；如果 $\Delta m_{ij} > 0$，$\Delta m_{il} = 0 (l \neq i)$，一旦我们把 $\boldsymbol{m}_i$ 规范化为单位向量，则 m_{il}减少。另一种可能的方

法是引进权衰减项(Oja 1982)，而(12.7)式的第二项就可以看作这样的项。Hertz，Krogh 和 Palmer(1991)更详细地讨论了竞争网络和 Hebbian 学习，并且展示这种网络如何学习做 PCA。Mao 和 Jain(1995)讨论了 PCA 和 LDA 的在线算法。

正如我们在第 7 章所看到的，一个问题是避免死中心，即存在却没有被实际利用的中心。在竞争网络，这对应因为被初始化远离任何输入而从来未能赢得竞争的中心。存在多种方法避免它：

1. 我们可以通过随机地选择输入实例来初始化 $\boldsymbol{m}_i$，并且确保它们从有数据的地方开始。

2. 我们可以使用领导者聚类算法并且逐个添加单元，总是将它们添加在需要它们的地方。一个例子是 ART 模型，我们将在 12.2.2 节讨论。

3. 更新时不仅更新最近单元的中心，而且也更新某些其他中心。随着它们被更新，它们也向输入移动，逐渐移向输入空间存在输入的部分，并最终赢得竞争。一个例子是我们将在 12.2.3 节讨论的 SOM。

4. 另一种可能引进良心(conscience)机制(Desieno 1988)：当前赢得竞争的单元有负罪感并允许其他单元获胜。

277
~
280

12.2.2 自适应共鸣理论

在计算参数之前，应当知道并指定分组数 k。另一种方法是增量的(incremental)，从单个分组开始，并在需要时添加新的分组。作为增量算法的一个例子，我们讨论自适应共鸣理论(adaptive resonance theory，ART)算法(Carpenter 和 Grossberg 1988)。在 ART 中，给定一个输入，所有的输出单元计算它们的值，并且选择与输入最相似的单元。如果使用如(12.6)式中的点积，则它是具有最大值的单元；如果使用欧氏距离，则它是具有最小值的单元。

让我们假设我们使用欧氏距离。如果最小值小于某个称作警戒值(vigilance)的阈值，则像在线 k-均值一样进行更新。如果距离大于警戒值，则增加一个新的输出单元，并且它的中心用该实例初始化。这定义了一个超球，其半径由定义每个单元范围体积的警戒值给定。当我们具有一个不能被任何单元覆盖的输入时，我们就增加一个新单元(参见图 12-4)。

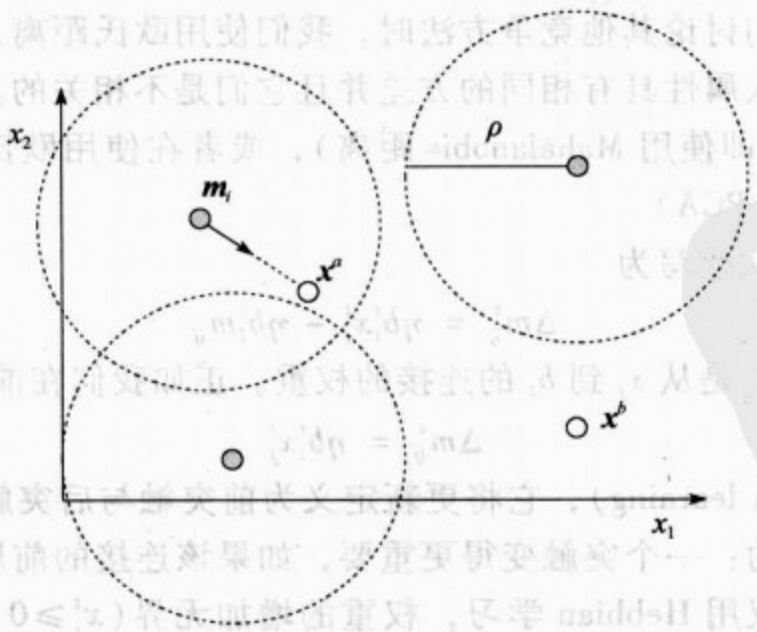


图 12-4 从 $\boldsymbol{x}^a$ 到最近中心的距离小于警戒值 ρ，中心像在线 k-均值一样进行更新。然而，$\boldsymbol{x}^b$ 与任何中心都不足够近，应当在该位置创建一个新的分组

记警戒值为 ρ，在每次更新我们使用下式：

$$b_i = \| \boldsymbol{m}_i - \boldsymbol{x}^t \| = \min_{l=1}^{k} \| \boldsymbol{m}_l - \boldsymbol{x}^t \| \tag{12.9}$$

$$\begin{cases} \boldsymbol{m}_{k+1} \leftarrow \boldsymbol{x}^t & \text{如果 } b_i > \rho \\ \Delta \boldsymbol{m}_i = \eta(\boldsymbol{x}^t - \boldsymbol{m}_i) & \text{否则} \end{cases}$$

在距离上设定阈值等价于在每个距离的重构误差上设定阈值，并且如果距离是欧氏距离、误差像(12.4)式那样定义，则这表明每个实例允许的最大重构误差为警戒值的平方。

12.2.3 自组织映射

避免死单元的一种方法是不仅更新获胜者，而且也更新某些其他单元。在 Kohonen (1990, 1995)提出的自组织映射(self-organizing map, SOM)中，单元下标(即，如 $\boldsymbol{m}_i$ 中的 i)定义单元的邻域(neighborhood)。当 $\boldsymbol{m}_i$ 是最近的中心时，除 $\boldsymbol{m}_i$ 更新之外，还更新它的近邻。例如，如果邻域大小为 2，则 $\boldsymbol{m}_{i-2}$，$\boldsymbol{m}_{i-1}$，$\boldsymbol{m}_{i+1}$，$\boldsymbol{m}_{i+2}$ 也更新，但是随邻域的加大，使用较小的权重。如果 i 是最近中心的下标，则中心按下式更新

$$\Delta \boldsymbol{m}_l = \eta e(l,i)(\boldsymbol{x}^t - \boldsymbol{m}_l) \tag{12.10}$$

其中 $e(l, i)$ 是邻域函数。当 $l = i$ 时 $e(l, i) = 1$，并随 $|l - i|$ 增大而减小。例如，定义它为高斯函数 $\mathcal{N}(i, \sigma)$：

$$e(l,i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{(l-i)^2}{2\sigma^2}\right] \tag{12.11}$$

为了收敛，邻域函数的支集随时间减小，例如 σ 减小，最终只有一个获胜者被更新。

由于邻域单元也向输入移动，我们避免了死单元，因为从它们的近邻朋友得到一点初始帮助之后，稍后的某个时候它们将赢得竞争(参见图 12-5)。

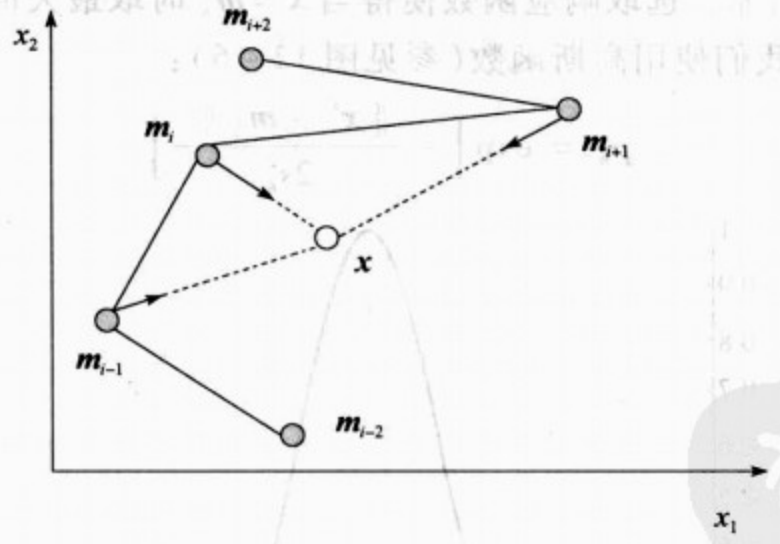


图 12-5 在 SOM 中，不仅最近的单元，而且还有它的近邻(就下标而言)都向输入移动。这里，邻域为 1；$\boldsymbol{m}_i$ 和它的 1-近邻被更新。注意，这里 $\boldsymbol{m}_{i+1}$ 远离 $\boldsymbol{m}_i$，但是随着它与 $\boldsymbol{m}_i$ 一起更新，并且当 $\boldsymbol{m}_{i+1}$ 是胜者时 $\boldsymbol{m}_i$ 也被更新，它们最终也成为输入空间的近邻

更新近邻具有如下效果：即使中心被随机初始化，因为它们一起朝着相同的输入移动，因此一旦系统收敛，具有相同邻近下标的单元也将是输入空间中的近邻。

在大部分应用中，单元被组织成二维映射(map)。即每个单元将具有两个下标 $\boldsymbol{m}_{i,j}$，并

且邻域定义在两个维上。如果 $m_{i,j}$ 是最近的中心，则中心按下式更新

$$\Delta m_{k,l} = \eta e(k,l,i,j)(x^t - m_{k,l}) \tag{12.12}$$

其中邻域函数现在是二维的。收敛后，这形成了原 d-维输入空间的二维地形图(topographical map)。该图包含了空间的高密度部分的许多单元，而对于没有输入的部分则不显示其中的任何单元。一旦该图收敛，则原空间靠近的输入被映射到该图中靠近的单元。从这种角度讲，该图可以解释为做一个非线性形式的多维缩放，将原来的 x 空间映射到二维(i, j)上。类似地，如果映射是一维的，则单元放置在输入空间最大密度的曲线上，作为主曲线(principal curve)。

281
~
283

12.3 径向基函数

在隐藏单元使用点积的多层感知器中(第 11 章)，每一个隐藏单元定义了一个超平面，并且使用 S 形非线性，隐藏单元具有 0 和 1 之间的值，对每个实例关于超平面的位置编码。每个超平面将输入空间一分为二，并且通常对于给定的输入，许多隐藏单元都具有非零输出。这称作分布表示(distributed representation)，因为输入被许多隐藏单元的同时激活重新编码。

另一种可能性是局部表示(local representation)，其中对于给定的输入，只有一个或几个单元是活跃的。就像这些局部调整(locally tuned)的单元在它们之间划分输入空间并且只对某些输入是有选择的。输入空间的单元中具有非零响应的部分称作接受域(receptive field)。输入空间则被这样的单元覆盖。

在大脑皮层的多处发现了具有这种响应特征的神经元。例如，视觉皮层细胞对刺激有选择地响应，既局部于视网膜的位置，又局部于视觉方向的角度。这种局部调整的细胞通常排列在大脑皮层图上，那里像在 SOM 中一样，细胞对其响应的变量值随它们在图中的位置而变化。

局部性意味有一个距离函数，度量给定输入 x 和单元 h 的位置 m_h 的相似度。通常，该度量取欧氏距离 $\| x - m_h \|$。选取响应函数使得当 $x = m_h$ 时取最大值，并且随着它们的相似性减小而减少。通常，我们使用高斯函数(参见图 12-6)：

$$p_h^t = \exp\left[-\frac{\| x^t - m_h \|^2}{2s_h^2} \right] \tag{12.13}$$

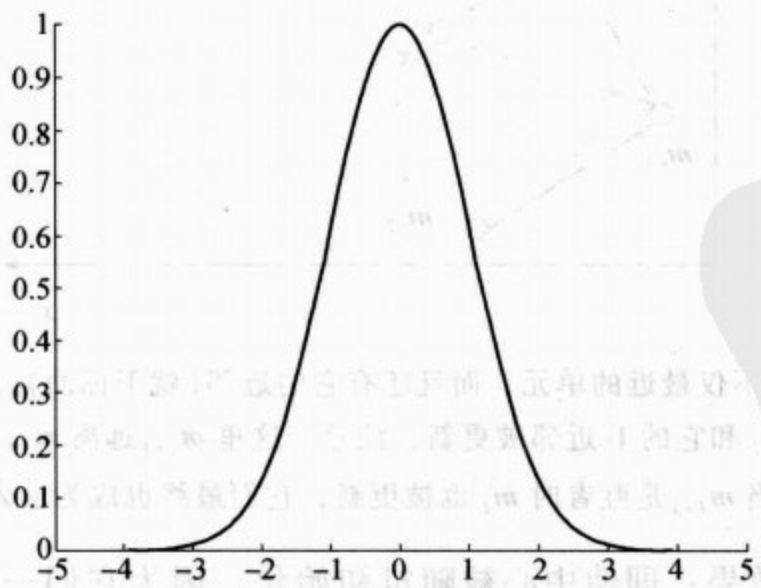
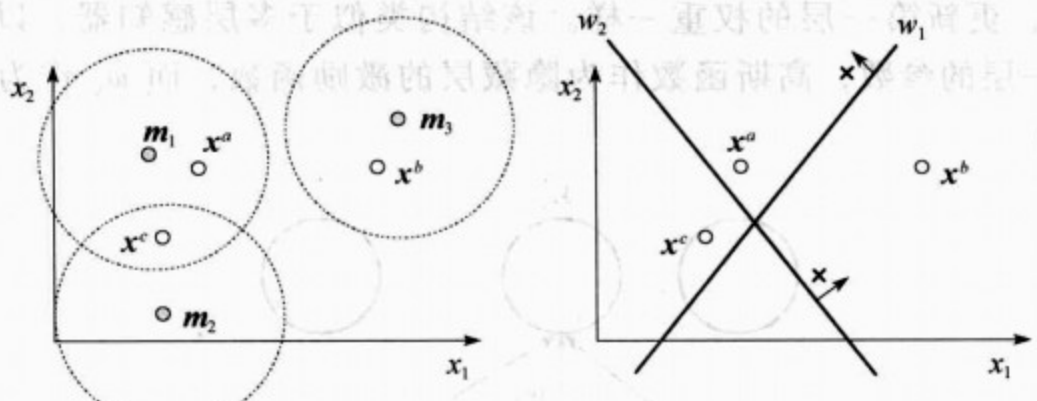


图 12-6 用于径向基函数网络的钟形函数的一维形式。这个函数 $m=0$，$s=1$。它像高斯函数但不是密度函数；其积分不等于 1。在($m-3s$, $m+3s$)中它非零，但是更保守的区间是($m-2s$, $m+2s$)

严格地说，这不是高斯密度，但是我们还是使用了相同的名字。$\boldsymbol{m}_j$ 和 s_j 分别表示局部单元 j 的中心和展宽，这样定义了一个径向对称的基函数。以使用更复杂的模型为代价，我们可以使用椭球，不同的维上具有不同的展宽，甚至使用 Mahalanobis 距离，允许相关的输入。

使用局部基函数的基本思想是，在输入数据中存在一些实例分组或簇，而对每个簇，我们定义一个基函数 p_h^t，当实例 $\boldsymbol{x}^t$ 属于簇 h 时它不为零。我们可以使用 12.2 节讨论的任意在线竞争方法来找出中心 $\boldsymbol{m}_h$。有一种简单、有效的启发式方法来找出展宽：一旦我们求出中心，我们就可以找出簇中的最远实例，并令 s_h 为它到簇中心距离的一半。我们本来也可以使用三分之一，但是我们宁愿保守一点。我们还可以使用统计聚类方法找出簇参数。例如，在高斯混合分布上使用第 7 章讨论的 EM 方法，找出簇参数均值、方差(和协方差)。

$p_h^t(h=1, \cdots, H)$定义了一个新的 H-维空间，并且形成 $\boldsymbol{x}^t$ 的新表示。我们也可以使用 b_h^t(12.2)式对输入编码，但是 b_h^t 为 0/1；p_h^t 具有附加的优点，它用(0，1)中的值对点到其中心的距离编码。该值衰减到 0 的速度依赖于 s_h。图 12-7 给出了一个例子，并且将这种局部表示与多层感知器使用的分布表示进行比较。由于高斯分布都是局部的，因此与使用分布表示相比，通常我们需要更多的局部单元，当输入是高维的时尤其如此。



空间(p_1，p_2，p_3)中的局部表示
$\boldsymbol{x}^a$:(1.0，0.0，0.0)
$\boldsymbol{x}^b$:(0.0，0.0，1.0)
$\boldsymbol{x}^c$:(1.0，1.0，0.0)

空间(h_1，h_2)中的分布表示
$\boldsymbol{x}^a$:(1.0，1.0)
$\boldsymbol{x}^b$:(0.0，1.0)
$\boldsymbol{x}^c$:(1.0，0.0)

图 12-7 局部表示与分布表示之间的差别。值是硬的 0/1 值。我们可以使用(0，1)之间的软值得到更多信息编码。在局部表示中，用高斯 RBF 做，它使用到中心 $\boldsymbol{m}_i$ 的距离；而在分布表示中，使用 S 形函数做，它使用到超平面 $\boldsymbol{w}_i$ 的距离

在监督学习的情况下，我们可以使用这种新的局部表示作为输入。如果我们使用感知器，则我们有

$$y^t = \sum_{h=1}^{H} w_h p_h^t + w_0 \tag{12.14}$$

其中 H 是基函数的个数。这种结构称作径向基函数(radial basis function，RBF)网络(Broomhead 和 Lowe 1988；Moody 和 Darken 1989)。通常，人们不使用多于一个高斯单元层的 RBF 网络。H 是复杂度参数，像多层感知器的隐藏单元数一样。之前，当它对应非监督学习中的中心数时，我们用 k 表示它。

这里，我们看到使用 p_h 而不是使用 b_h 的优点。由于 b_h 是 0/1，如果(12.14)式使用 b_h 而不是 p_h，则它将给出分段常量近似，在单元区域的边界不连续。p_h 值是软的并导致光滑的近似，从一个区域到另一个时取加权平均。我们可以容易地看到这种网络是一种通用逼近，因为

284 ≀ 286

给定足够多的单元，它可以以期望的精度逼近任意函数。对于我们期望的精度，我们可以形成一个输入空间的网格，实际上对每个网格定义一个单元，并设置它的权重 w_h 为期望的输出值。

这种结构与非参数估计(例如我们在第 8 章所看到的 Parzen 窗口)很相似，并且 p_h 可以看作核函数。不同之处在于我们没有整个训练集上的核函数，而是使用聚类方法将它们分组，使用更少的核函数。单元数 H 是复杂度参数，在简洁性和准确性之间平衡。使用更多单元，我们就更好地逼近训练数据，但是我们得到更复杂的模型并有过分拟合的风险；太少可能拟合不足。最佳值仍然用交叉确认确定。

一旦给定和固定 $\boldsymbol{m}_h$ 和 s_h，p_h 也是固定的。然后可以容易地批处理或在线地训练 w_h。对于回归，这是一个线性回归模型(p_h 作为输入)，并且 w_h 可以解析地求解，而不需要迭代(4.6 节)。对于分类，我们需要借助于迭代过程。我们在第 10 章讨论过这些学习方法，此处不再赘述。

这里，我们要做的是一个两阶段过程：我们使用非监督方法确定中心，然后在其上构建监督层。这称作混合学习(hybrid learning)。我们还可以用监督的方式学习所有的参数，包括 $\boldsymbol{m}_h$ 和 s_h。(12.13)式的径向基函数是可微的，并且我们可以后向传播，就像我们在多层感知器后向传播，更新第一层的权重一样。该结构类似于多层感知器，以 p_h 为隐藏单元，$\boldsymbol{m}_h$ 和 s_h 作为第一层的参数，高斯函数作为隐藏层的激励函数，而 w_h 作为第二个隐藏层的权重(见图 12-8)。

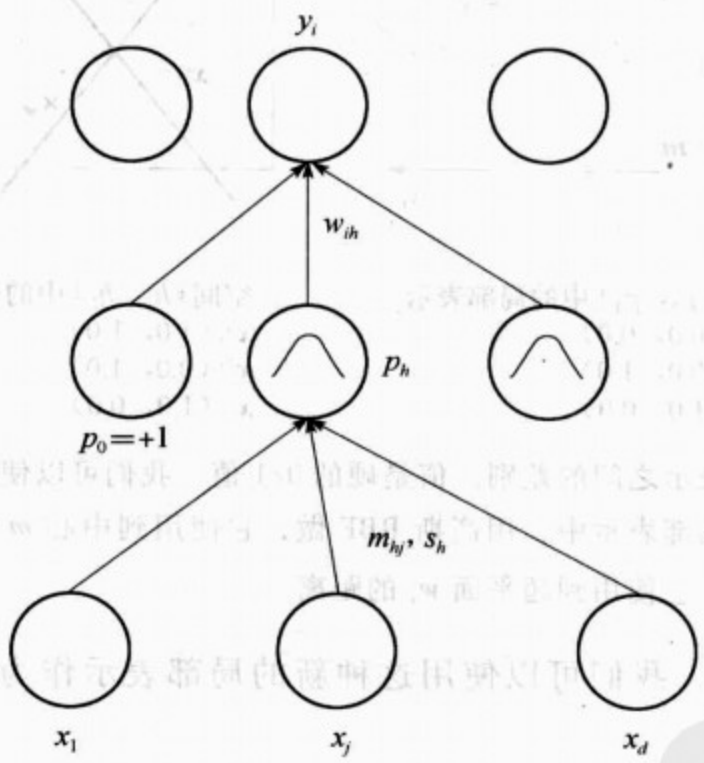


图 12-8 RBF 网络，其中 p_h 是使用钟形激励函数的隐藏单元，$\boldsymbol{m}_h$，s_h 是第一层的参数，而 w_i 是第二层的权重

但是，在我们讨论这些之前，我们应当注意到训练两层网络很慢。混合模型一次训练一层，因而比较快。另一种技术，称作锚(anchor)方法，将中心设置为从训练集随机选取的模式，而不进一步更新。如果有许多单元，这足以满足需要。

另一方面，精度通常没有使用完全监督方法高。考虑输入是均匀分布的情况。k-均值聚类均匀地安放单元。如果函数在一小部分空间稍有变化，则更好的想法是将更多单元安放在

函数变化快的地方，以便使误差尽可能小；这正是完全监督方法所要做的。

让我们讨论如何在完全监督方式下训练所有参数。方法与用于多层感知器的后向传播一样。让我们考虑具有多个输出的回归。批处理的误差为

$$E(\{\boldsymbol{m}_h, s_h, w_{ih}\}_{i,h} | \mathcal{X}) = \frac{1}{2}\sum_t \sum_i (r_i^t - y_i^t)^2 \tag{12.15}$$

其中

$$y_i^t = \sum_{h=1}^{H} w_{ih} p_h^t + w_{i0} \tag{12.16}$$

使用梯度下降，我们得到第二层权重的如下更新规则：

$$\Delta w_{ih} = \eta \sum_t (r_i^t - y_i^t) p_h^t \tag{12.17}$$

这是通常的感知器更新规则，其中 p_h 作为输入。通常，p_h 交叠不多，并且在每次迭代中，只有少量的 p_h 非零，并且只有它们的 w_h 被更新。这就是为什么 RBF 网络学习非常快，并且比使用分布表示的多层感知器快的原因。

类似地，我们可以用后向传播(链规则)得到中心和展宽的更新方程：

$$\Delta m_{hj} = \eta \sum_t \left[\sum_i (r_i^t - y_i^t) w_{ih}\right] p_h^t \frac{(x_j^t - m_{hj})}{s_h^2} \tag{12.18}$$

$$\Delta s_h = \eta \sum_t \left[\sum_i (r_i^t - y_i^t) w_{ih}\right] p_h^t \frac{\|\boldsymbol{x}^t - \boldsymbol{m}_h\|^2}{s_h^3} \tag{12.19}$$

让我们比较(12.18)和(12.5)式：首先，这里我们使用 p_h 而不是 b_h，这意味不仅最近的单元而且所有的单元都根据它们的中心和展宽而被更新。其次，这里更新是监督的，并且包含后向传播的误差项。更新不仅依赖于输入，而且依赖于最终的误差 $(r_i^t - y_i^t)$、单元对输出的影响 w_{ih}、单元的活性 p_h 和输入 $(x - \boldsymbol{m}_i)$。

对于分类，我们有

$$y_i^t = \frac{\exp\left[\sum_h w_{ih} p_h^t + w_{i0}\right]}{\sum_k \exp\left[\sum_h w_{kh} p_h^t + w_{k0}\right]} \tag{12.20}$$

而互熵误差为

$$E(\{\boldsymbol{m}_h, s_h, w_{ih}\}_{i,h} | \mathcal{X}) = -\sum_t \sum_i r_i^t \log y_i^t \tag{12.21}$$

使用梯度下降，可以类似地导出更新规则(习题 2)。

让我们再考虑(12.14)式：对于任意输入，如果 p_h 非零，则它对输出的贡献为 w_h。它的贡献是常量拟合，由 w_h 给定。通常，高斯函数交叠不多，并且它们之中的一或两个具有非零的 p_h 值。在任何情况下，只有少数单元对输出有贡献。w_0 是偏移常量，加到活跃(非零)单元的加权和上。我们还看到如果所有的 p_h 均为 0，则 $y = w_0$。这样，我们可以把 w_0 看作是 y 的缺省值：如果没有高斯单元是活跃的，则输出由该值给定。因此，有可能使得该“默认模型”具有更强的能力。例如，我们可以令

$$y^t = \sum_{h=1}^{H} w_h p_h^t + \boldsymbol{v}^T \boldsymbol{x}^t + v_0 \tag{12.22}$$

在这种情况下，默认模型是线性的：$\boldsymbol{v}^T\boldsymbol{x}' + v_0$。当它们非零时，高斯模型好像是“例外”并修改输出，补偿期望输出与默认输出之差。这种模型可以以监督方式训练，而默认模型与 w_h 一起训练(习题 3)。

12.4 结合基于规则的知识

如果我们能够结合先验知识(prior knowledge)来初始化，则任何学习系统的训练都可以更简单。例如，先验知识可以以一组规则的形式提供，指定该模型(例如 RBF 网络)必须学习的输入/输出映射。这种情况在业界和医学应用中经常出现，在应用中规则可以由专家提供。类似地，一旦网络被训练，就可以从中提取规则，使得问题的解更容易理解。

包含先验知识还有其他优点。如果需要将网络外推到输入空间中从未见到训练数据的区域，可以依赖这种先验知识。此外，在许多控制应用中，需要网络一开始就做出合理的预测。在它看到足够多的训练数据之前，必须主要依赖这种先验知识。

在许多应用中，我们通常被告知一些基本规则，开始需要遵循，而后通过经验来精炼和改变。我们关于问题的初始知识越好，我们得到好性能就越快，并且需要的训练就越少。

使用 RBF 网络，这种包含先验知识或提取学习的知识很容易做，因为单元是局部的。这使得规则提取(rule extraction)更容易(Tresp、Hollatz 和 Ahmad 1997)。一个例子是

290

$$\mathsf{IF}((x_1 \approx a)\mathsf{AND}(x_2 \approx b))\mathsf{OR}(x_3 \approx c)\mathsf{THEN}\ y = 0.1 \tag{12.23}$$

其中 $x_1 \approx a$ 意指“x_1 约等于 a”。在 RBF 框架中，这个规则被两个高斯单元编码为

$$p_1 = \exp\left[-\frac{(x_1 - a)^2}{2s_1^2}\right] \cdot \exp\left[-\frac{(x_2 - b)^2}{2s_2^2}\right], \quad \text{其中 } w_1 = 0.1$$

$$p_2 = \exp\left[-\frac{(x_3 - c)^2}{2s_3^2}\right], \quad \text{其中 } w_2 = 0.1$$

“约等于”被一个高斯函数建模，这里中心是理想值，展宽表示理想值周围允许的差。合取是两个一元高斯函数的积，是二元高斯函数。于是，第一个乘积项可以被二维(即 $\boldsymbol{x} = [x_1, x_2]$)高斯函数处理，其中心在$(a, b)$，而在两个维上的展宽由 s_1 和 s_2 给定。析取被两个单独的高斯函数建模，每个处理一个析取项。

给定标记的训练数据，使用较小的 η 值，这样构造的 RBF 网络的参数在初始构造后可以微调。

这种表示方法与模糊逻辑方法有关，那里(12.23)式称作模糊规则(fuzzy rule)。检查近似相等的高斯基函数对应模糊隶属关系函数(fuzzy membership function)(Berthold 1999；Cherkassky 和 Mulier 1998)。

12.5 规范化基函数

在(12.14)式中，对于一个输入，可能所有的 p_h 都为 0。在某些应用中，我们可能希望有一个规范化步骤，确保局部单元值的和为 1，从而确保对于任何输入，至少存在一个非零单元：

$$g_h^t = \frac{p_h^t}{\sum_{l=1}^{H} p_l^t} = \frac{\exp[-\|\boldsymbol{x}^t - \boldsymbol{m}_h\|^2/2s_h^2]}{\sum_l \exp[-\|\boldsymbol{x}^t - \boldsymbol{m}_l\|^2/2s_l^2]} \tag{12.24}$$

图 12-9 给出了一个例子。取 p_h 为 $p(\boldsymbol{x}|h)$，g_h 对应 $\boldsymbol{x}$ 属于单元 h 的后验概率 $p(h|\boldsymbol{x})$。它就像单元在它们之间划分输入空间。我们可以想象 g_h 本身是分类器，为给定的输入选择响应单元。这种分类基于距离来做，就像在有参高斯分类器中那样(第 5 章)。

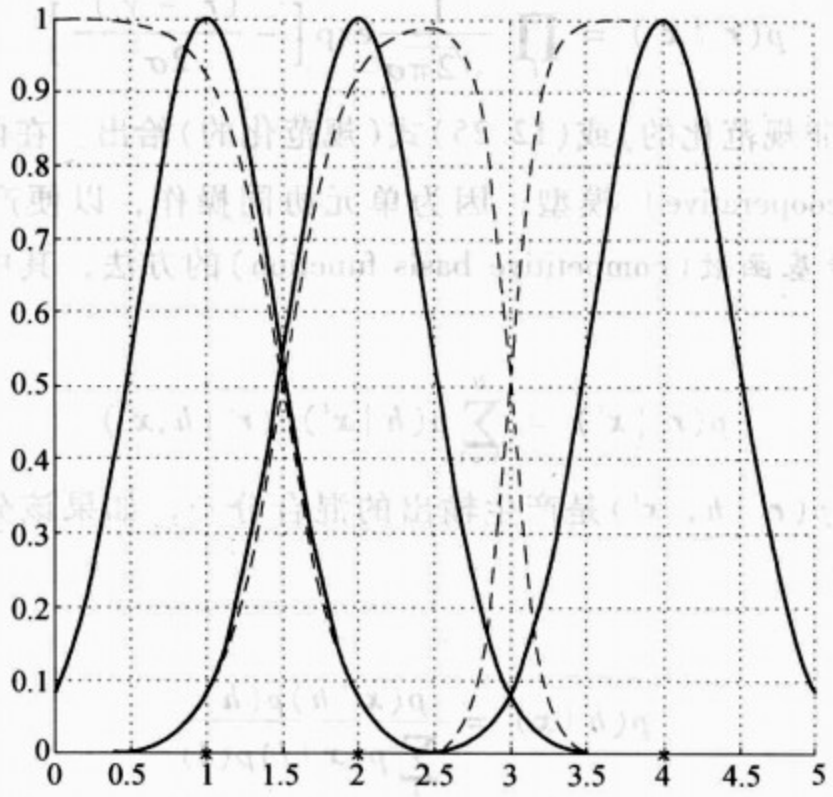


图 12-9 规范化前(－)和规范化后(－－)的三个高斯分布，其中心用“＊”标记。注意一个单元的非零区域还依赖其他单元的位置。如果展宽较小，则规范化实现较硬的划分；使用较大的展宽，单元重叠更多

输出是加权和

$$y_i^t = \sum_{h=1}^{H} w_{ih} g_h^t \tag{12.25}$$

这里不需要偏倚项，因为对于每个 $\boldsymbol{x}$，至少有一个非零的 g_h。使用 g_h 而不是 p_h 并不引进附加的参数；它只是将单元联系在一起：p_h 仅依赖于 $\boldsymbol{m}_h$ 和 s_h，但是由于规范化，g_h 依赖于所有单元的中心和展宽。

对于回归，使用梯度下降，我们有如下更新规则：

$$\Delta w_{ih} = \eta \sum_t (r_i^t - y_i^t) g_h^t \tag{12.26}$$

$$\Delta m_{hj} = \eta \sum_t \sum_i (r_i^t - y_i^t)(w_{ih} - y_i^t) g_h^t \frac{(x_j^t - m_{hj})}{s_h^2} \tag{12.27}$$

可以类似地导出 s_h 的更新规则和用于分类的规则。让我们把这些规则与具有非规范化高斯分布的 RBF 的规则(12.17)式进行比较。这里，我们使用 g_h 而不是 p_h，这使得单元的更新不仅依赖于它自己的参数，而且也依赖于其他单元的中心和展宽。比较(12.27)和(12.18)式，我们看到我们有 $(w_{ih} - y_i^t)$ 而不是 w_{ih}，这展示了规范化在输出上的作用。“负责任”的单元希望降低它的输出 w_{ih} 和最终输出 y_i^t 之间的差，正比于它的责任 g_h。

291
≀
292

12.6 竞争的基函数

正如我们迄今为止所看到的，在 RBF 网络中，最终的输出被局部单元贡献的加权和所确定。尽管单元是局部的，但是重要的是最终的加权和，并且我们希望使它与要求的输出尽可能接近。例如对于回归，我们最小化(12.15)式，这基于概率模型

$$p(\boldsymbol{r}^t \mid \boldsymbol{x}^t) = \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left[- \frac{(r_i^t - y_i^t)^2}{2\sigma^2} \right] \tag{12.28}$$

其中 y_i^t 由(12.16)式(非规范化的)或(12.25)式(规范化的)给出。在两种情况下，我们都可以将模型看作协同 (cooperative) 模型，因为单元协同操作，以便产生最终的输出 y_i^t。现在，我们讨论使用竞争基函数(competitive basis function)的方法，其中我们假定输出取自混合模型

$$p(\boldsymbol{r}^t \mid \boldsymbol{x}^t) = \sum_{h=1}^{H} p(h \mid \boldsymbol{x}^t) p(\boldsymbol{r}^t \mid h, \boldsymbol{x}^t) \tag{12.29}$$

$p(h \mid \boldsymbol{x}^t)$是混合比例，$p(\boldsymbol{r}^t \mid h, \boldsymbol{x}^t)$是产生输出的混合分支，如果该分支被选择的话。注意这两项都依赖于输入 $\boldsymbol{x}$。

混合比例为

293

$$p(h \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid h) p(h)}{\sum_l p(\boldsymbol{x} \mid l) p(l)} \tag{12.30}$$

$$g_h^t = \frac{a_h \exp[- \| \boldsymbol{x}^t - \boldsymbol{m}_h \|^2 / 2s_h^2]}{\sum_l a_l \exp[- \| \boldsymbol{x}^t - \boldsymbol{m}_l \|^2 / 2s_l^2]} \tag{12.31}$$

一般，我们假定 a_h 相等并忽略它们。让我们先考虑回归，其中分支是高斯的。在(12.28)式，噪声被加到加权和上；这里一个分支被选中，并且噪声加到它的输出 y_{ih}^t上。

使用(12.29)式的混合模型，对数似然是

$$\mathcal{L}(\{\boldsymbol{m}_h, s_h, w_{ih}\}_{i,h} \mid \mathcal{X}) = \sum_t \log \sum_h g_h^t \exp\left[- \frac{1}{2} \sum_i (r_i^t - y_{ih}^t)^2 \right] \tag{12.32}$$

其中 $y_{ih}^t = w_{ih}$是由分支 h 对输出 i 做的常量拟合。严格地说，它不依赖于 $\boldsymbol{x}$。(在 12.8.2 节，我们讨论竞争的混合专家模型，其中局部拟合是 $\boldsymbol{x}$ 的线性函数。)我们看到如果 g_h^t 为 1，则它对产生正确的输出负责，并且需要最小化它的预测误差的平方和 $\sum_i (r_i^t - y_{ih}^t)^2$。

使用梯度上升最大化该对数似然，我们得到

$$\Delta w_{ih} = \eta \sum_t (r_i^t - y_{ih}^t) f_h^t \tag{12.33}$$

其中

$$f_h^t = \frac{g_h^t \exp\left[- \frac{1}{2} \sum_i (r_i^t - y_{ih}^t)^2 \right]}{\sum_l g_l^t \exp\left[- \frac{1}{2} \sum_i (r_i^t - y_{il}^t)^2 \right]} \tag{12.34}$$

$$p(h \mid \boldsymbol{r}, \boldsymbol{x}) = \frac{p(h \mid \boldsymbol{x}) p(\boldsymbol{r} \mid h, \boldsymbol{x})}{\sum_l p(l \mid \boldsymbol{x}) p(\boldsymbol{r} \mid l, \boldsymbol{x})} \tag{12.35}$$

$g_h^t \equiv p(h \mid \boldsymbol{x}^t)$是给定输入、单元 h 的后验概率，并且它依赖于所有单元的中心和展宽。$f_h^t \equiv p(h \mid \boldsymbol{r}, \boldsymbol{x}^t)$是给定输入和期望的输出、单元 h 的后验概率，也在选择负责单元时考虑误差。

类似地，我们可以推导更新中心的规则：

$$\Delta m_{hj} = \eta \sum_t (f_h^t - g_h^t) \frac{(x_j^t - m_{hj})}{s_h^2} \tag{12.36}$$

f_h 是单元 h 的后验概率，也考虑要求的输出，而 g_h 是仅使用输入空间信息的后验概率。它们的差是中心的误差项。可以类似地导出 Δs_h。在协同情况下，并不强求单元是局部的。为了降低误差，均值和展宽都可以取任意值；有时，甚至可以增加和展平展宽。然而，在竞争情况下，为了提高似然，单元必须是局部的，它们之间更加分离，并具有更小的展宽。 294

对于分类，每个分支本身是多项式。于是，对数似然为

$$\mathcal{L}(\{\boldsymbol{m}_h, s_h, w_{ih}\}_{i,h} \mid \mathcal{X}) = \sum_t \log \sum_h g_h^t \prod_i (y_{ih}^t)^{r_i^t} \tag{12.37}$$

$$= \sum_t \log \sum_h g_h^t \exp\left[\sum_i r_i^t \log y_{ih}^t\right] \tag{12.38}$$

其中

$$y_{ih}^t = \frac{\exp w_{ih}}{\sum_k \exp w_{kh}} \tag{12.39}$$

可以使用梯度上升导出 w_{ih}、$\boldsymbol{m}_h$ 和 s_h 的更新规则，这包括

$$f_h^t = \frac{g_h^t \exp\left[\sum_i r_i^t \log y_{ih}^t\right]}{\sum_l g_l^t \exp\left[\sum_i r_i^t \log y_{il}^t\right]} \tag{12.40}$$

在第 7 章，我们讨论了用混合高斯模型拟合数据的 EM 算法。也可以将 EM 推广到监督学习。实际上，计算 f_h^t 对应 E 步。$f_h^t \equiv p(\boldsymbol{r} \mid h, \boldsymbol{x}^t)$取代了 $p(h \mid \boldsymbol{x}^t)$，后者是应用处于非监督状态时我们在第 7 章的 E 步所使用的。对于回归，在 M 步我们用下式更新参数

$$\boldsymbol{m}_h = \frac{\sum_t f_h^t \boldsymbol{x}^t}{\sum_t f_h^t} \tag{12.41}$$

$$s_h = \frac{\sum_t f_h^t (\boldsymbol{x}^t - \boldsymbol{m}_h)(\boldsymbol{x}^t - \boldsymbol{m}_h)^T}{\sum_t f_h^t} \tag{12.42}$$

$$w_{ih} = \frac{\sum_t f_h^t r_i^t}{\sum_t f_h^t} \tag{12.43}$$

我们看到 w_{ih}是加权平均，其中权重是给定输入和期望的输出、单元的后验概率。对于分类，M 步没有解析解并且需要借助于迭代过程，例如梯度上升(Jordan 和 Jacobs 1994)。 295

12.7 学习向量量化

假设对每个类我们有 H 个单元，已经被这些类标记。这些单元已经被它们的类中的实例随机初始化。在每次迭代中，我们发现单元 $\boldsymbol{m}_i$ 在欧式距离下最接近输入实例，并使用如下更新规则：

$$\begin{cases}\Delta \boldsymbol{m}_i = \eta(\boldsymbol{x}^t - \boldsymbol{m}_i) & \text{如果 } \boldsymbol{x}^t \text{ 和 } \boldsymbol{m}_i \text{ 具有相同的类标号} \\ \Delta \boldsymbol{m}_i = -\eta(\boldsymbol{x}^t - \boldsymbol{m}_i) & \text{否则}\end{cases} \tag{12.44}$$

如果最近的中心具有正确的标号，则它将移向输入，以便更好地代表它。如果它属于错误的类，则它远离输入，我们期望的情形是，如果它移得足够远，则在未来的迭代中正确的类将是最近的。这称作学习向量量化(learning vector quantization，LVQ)模型，由 Kohonen 提出(1990，1995)。

LVQ 更新方程类似于(12.36)式，其中中心移动的方向依赖于两个值的差：获胜单元基于输入距离的预测和获胜者基于要求的输出。

12.8 混合专家模型

在 RBF 中，对应每个局部小片，我们给出一个常量拟合。在对于任意输入，我们有一个 g_h 为 1，而其余为 0 的情况下，我们得到一个分段常量近似，其中对于输出 i，小片 h 的局部拟合由 w_{ih} 给出。从泰勒展开式我们知道在每个点，函数可以写成

$$f(x) = f(a) + (x - a)f'(a) + \cdots \tag{12.45}$$

这样，如果 x 足够接近 a 并且 $f'(a)$ 接近 0，即如果 $f(x)$ 在 a 附近是平坦的，则常量近似很好。如果不是这种情况，则我们需要将空间划分成大量小片。当输入维度很高时，由于维灾难的缘故，这将是一个特别严重的问题。

一种可供选择的方法是考虑泰勒展开式的下一项(即线性项)，使用分段线性近似(piecewise linear approximation)。这就是混合专家模型(mixture of experts)所做的(Jacobs 等 1991)。我们令

296

$$y_i^t = \sum_{h=1}^{H} w_{ih} g_h^t \tag{12.46}$$

它与(12.25)式一样，但是这里小片 h 对输出 i 的贡献 w_{ih} 不是常量，而是输入的线性函数：

$$w_{ih}^t = \boldsymbol{v}_{ih}^T \boldsymbol{x}^t \tag{12.47}$$

$\boldsymbol{v}_{ih}$ 是参数向量，定义线性函数，并包含一个偏倚项，使得混合专家模型是 RBF 网络的推广。单元活性可以取规范化的 RBF：

$$g_h^t = \frac{\exp[-\|\boldsymbol{x}^t - \boldsymbol{m}_h\|^2 / 2s_h^2]}{\sum_l \exp[-\|\boldsymbol{x}^t - \boldsymbol{m}_l\|^2 / 2s_l^2]} \tag{12.48}$$

除第二层权重不是常量而是线性模型的输出外，这可以看作 RBF 网络(参见图 12-10)。Jacobs 等(1991)用另一种方法来看它：他们将 w_h 看作线性模型，每个都取输入，并称它们为专家。g_h 被看作一个门网络(gating network)的输出。门网络就像其输出之和为 1 的分类器一样，将输入指派给一个专家(参见图 12-11)。

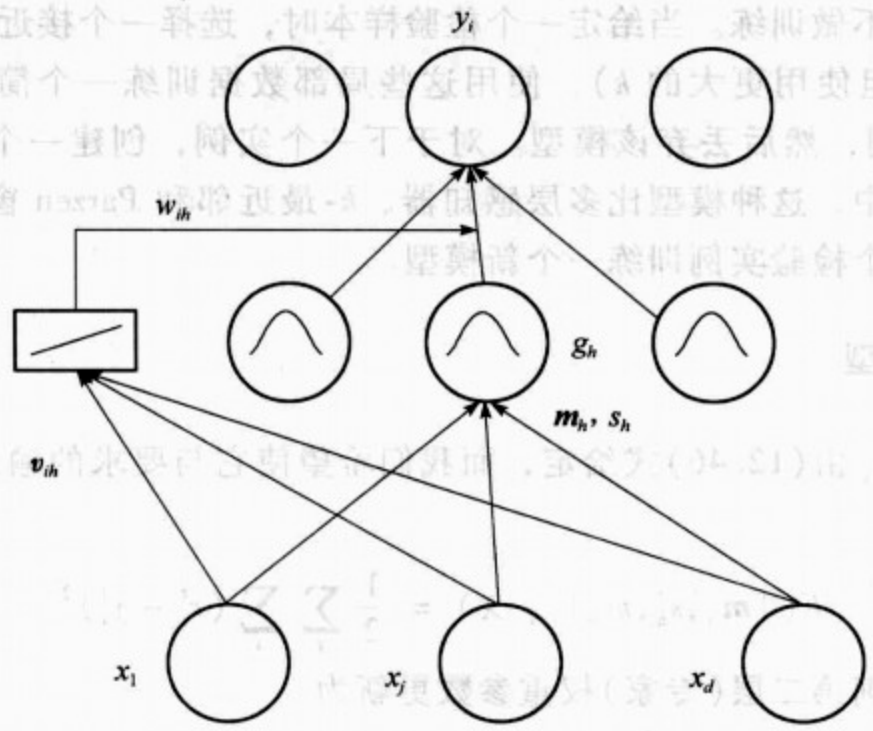


图 12-10 混合专家模型可以看作 RBF 网络，其中第二层的权重是线性模型的输出。为清晰起见，只显示了一个线性模型

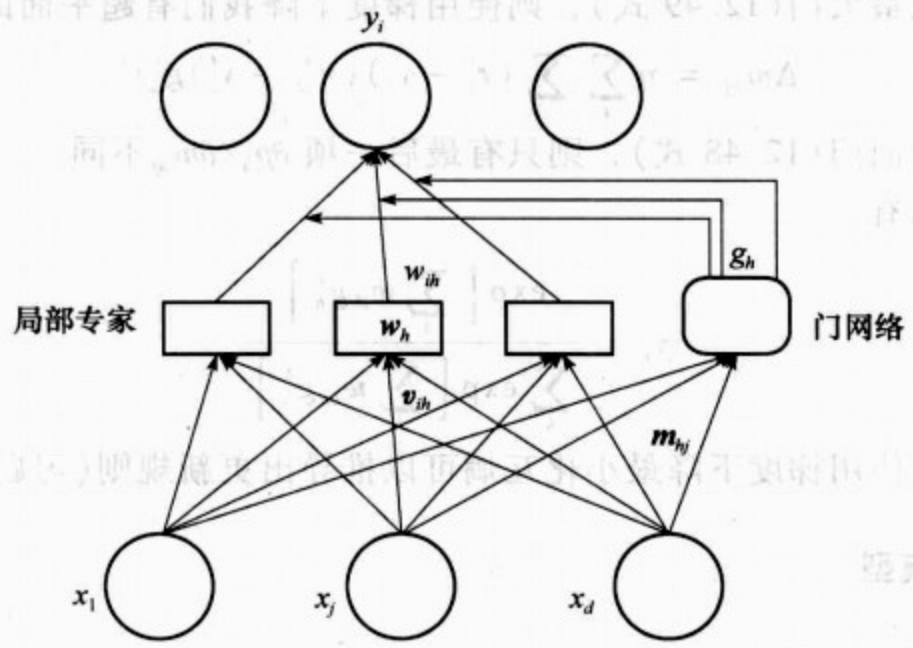


图 12-11 混合专家模型可以看作组合多种模型的模型。w_h 是模型，而门网络是另一种确定每个模型的权重的模型，如 g_h 所示。从这个角度来看，专家和门网络都不限于是线性的

用这种方式看待门网络，任何分类器都可以用在门网络之中。当 $\boldsymbol{x}$ 是高维的时，使用局部高斯单元可能需要大量专家，而 Jacobs 等(1991)提议取

$$g_h^t = \frac{\exp[\boldsymbol{m}_h^T \boldsymbol{x}^t]}{\sum_l \exp[\boldsymbol{m}_l^T \boldsymbol{x}^t]} \tag{12.49}$$

这是一个线性分类器。注意，$\boldsymbol{m}_h$ 不再是中心，而是超平面，并因此包含偏倚值。门网络实现了分类，它将输入区域线性地划分成专家 h 负责的区域和其他专家负责的区域。正如我们将在第 15 章再次看到的，混合专家模型是一种组合多个模型的一般结构；专家和门网络都可以是非线性的，例如，包含多层感知器而不是线性感知器(习题 5)。

Bottou 和 Vapnik(1992)提出了一种类似于混合专家模型并进行线性光滑的结构。在他们

297 ~ 298

的方法中，最初时并不做训练。当给定一个检验样本时，选择一个接近检验实例的数据子集（与 k-最近邻一样，但使用更大的 k），使用这些局部数据训练一个简单模型（如线性分类器），对实例做出预测，然后丢弃该模型。对于下一个实例，创建一个新模型，如此下去。在手写数字识别应用中，这种模型比多层感知器、k-最近邻和 Parzen 窗口具有更小的误差。缺点是需要现场对每个检验实例训练一个新模型。

12.8.1 协同专家模型

在协同情况下，y_i^t 由(12.46)式给定，而我们希望使它与要求的输出 r_i^t 尽可能接近。对于回归，误差函数是

$$E(\{\boldsymbol{m}_h, s_h, \boldsymbol{w}_{ih}\}_{i,h} | \mathcal{X}) = \frac{1}{2}\sum_t \sum_i (r_i^t - y_i^t)^2 \tag{12.50}$$

使用梯度下降，则第二层（专家）权重参数更新为

$$\Delta \boldsymbol{v}_{ih} = \eta \sum_t (r_i^t - y_i^t) g_h^t \boldsymbol{x}^t \tag{12.51}$$

与(12.26)式比较，我们看到唯一的区别是，新的更新是输入的函数。

如果我们使用软最大门(12.49 式)，则使用梯度下降我们有超平面的如下更新规则：

$$\Delta m_{hj} = \eta \sum_t \sum_i (r_i^t - y_i^t)(w_{ih}^t - y_i^t) g_h^t x_j^t \tag{12.52}$$

如果我们使用径向门(12.48 式)，则只有最后一项 $\partial p_h / \partial m_{hj}$ 不同。

对于分类，我们有

$$y_i = \frac{\exp\left[\sum_h w_{ih} g_h^t\right]}{\sum_k \exp\left[\sum_h w_{kh} g_h^t\right]} \tag{12.53}$$

299 其中 $w_{ih} = \boldsymbol{v}_{ih}^T \boldsymbol{x}$，并且使用梯度下降最小化互熵可以推导出更新规则（习题 6）。

12.8.2 竞争专家模型

正如竞争的 RBF，我们有

$$\mathcal{L}(\{\boldsymbol{m}_h, s_h, \boldsymbol{w}_{ih}\}_{i,h} | \mathcal{X}) = \sum_t \log \sum_h g_h^t \exp\left[-\frac{1}{2}\sum_i (r_i^t - y_{ih}^t)^2\right] \tag{12.54}$$

其中 $y_{ih}^t = w_{ih}^t = \boldsymbol{v}_{ih} \boldsymbol{x}^t$。使用梯度上升，我们得到

$$\Delta \boldsymbol{v}_{ih} = \eta \sum_t (r_i^t - y_{ih}^t) f_h^t \boldsymbol{x}^t \tag{12.55}$$

$$\Delta \boldsymbol{m}_h = \eta \sum_t (f_h^t - g_h^t) \boldsymbol{x}^t \tag{12.56}$$

假定软最大门如(12.49)式。

对于分类，我们有

$$\mathcal{L}(\{\boldsymbol{m}_h, s_h, \boldsymbol{w}_{ih}\}_{i,h} | \mathcal{X}) = \sum_t \log \sum_h g_h^t \prod_i (y_{ih}^t)^{r_i^t} \tag{12.57}$$

$$= \sum_t \log \sum_h g_h^t \exp\left[\sum_i r_i^t \log y_{ih}^t\right] \tag{12.58}$$

其中

$$y_{ih}^t = \frac{\exp w_{ih}^t}{\sum_k \exp w_{kh}^t} = \frac{\exp[\boldsymbol{v}_{ih}\boldsymbol{x}^t]}{\sum_k \exp[\boldsymbol{v}_{kh}\boldsymbol{x}^t]} \tag{12.59}$$

Jordan 和 Jacobs(1994)将 EM 推广到具有局部线性模型的竞争情况。Alpaydın 和 Jordan (1996)比较了用于分类人物的协同和竞争模型，发现协同模型一般更精确，但是竞争版本学习更快。这是因为在协同情况下，重叠更多并且实现了比较光滑的近似，因此更适合回归问题。竞争模型做比较硬的划分；通常，对于一个输入，只有一个专家是活跃的，因此学习更快。

12.9 层次混合专家模型

在图 12-11 中，我们看到一组专家和一个选择一个专家作为输入的函数的门网络。在*层次混合专家模型*(hierarchical mixture of expert)中，我们以递归的方式用一个完整的混合专家系统取代每个专家(Jordan 和 Jacobs 1994)。这种结构可以看作是一棵决策树(第 9 章)，其 300
中门网络可以看作决策结点。当门网络是线性的时，这很像第 9.6 节讨论的线性多元决策树。区别是门网络并不做硬决策，而是取来自子女贡献的加权和。叶结点是线性模型，并且它们的决策被取平均值并在树中向上传播。树根给出最终的输出，输出是所有树叶决策的加权和。这是一棵*软决策树*(soft decision tree)，与我们以前看到的决策树不同，其中只取一条从树根到树叶的路径。

一旦结构选定，即选定深度、专家和门模型，整棵树就可以从标记的样本中学习。Jordan 和 Jacobs(1994)为这样的结构推导出了梯度下降和 EM 学习规则。

12.10 注释

RBF 网络可以看作神经网络，由简单处理单元的网络实现。它不同于多层感知器，第一层和第二层实现了不同的函数。Omohundro(1987)讨论了如何用神经网络实现局部模型，并且还提出了相关局部单元快速局部化的层次数据结构。Specht(1991)证明 Parzen 窗口可以作为神经网络实现。

Platt(1991)提出了 RBF 的增量版本，那里新单元可以在必要时添加。类似地，Fritzke (1995)提出了 SOM 的增长版本。

Lee(1991)在手写数字识别应用上比较了 k-最近邻、多层感知器和 RBF 网络，并且结论是三种方法都具有小误差率。RBF 网络学习比多层感知器上的后向传播快，但是使用更多参数。就分类速度和存储需求而言，这两种方法都优于 k-NN。在实际应用中，像时间、存储量和计算复杂性等实际限制可能比误差率的些许差别更重要。

Kohonen 的 SOM(1990，1995)是最流行的神经网络方法之一，已经用于各种各样的应用，包括探测式数据分析和作为监督学习之前的预处理步骤。一个有趣和成功应用的例子是 301
旅行商人的问题(Angeniol、Vaubois 和 Le Texier 1988)。

12.11 习题

1. 给出一个实现 XOR 的 RBF 网络。
2. 为分类的 RBF 网络推导更新方程(12.20 和 12.21 式)。
3. 展示如何训练(12.22)式给定的系统。
4. 比较混合专家结构和 RBF 网络的参数个数。
5. 给出混合专家结构的公式，其中专家和门网络都是多层感知器。为回归和预测推导更新方程。
6. 为分类推导协同混合专家模型的更新方程。
7. 为分类推导竞争混合专家模型的更新方程。
8. 给出具有两个层次混合专家结构的公式。使用梯度下降，为回归和分类推导更新方程。

12.12 参考文献

Alpaydın, E., and M. I. Jordan. 1996. "Local Linear Perceptrons for Classification." *IEEE Transactions on Neural Networks* 7: 788 – 792.

Angeniol, B., G. Vaubois, and Y. Le Texier. 1988. "Self Organizing Feature Maps and the Travelling Salesman Problem." *Neural Networks* 1: 289 – 293.

Berthold, M. 1999. "Fuzzy Logic." In *Intelligent Data Analysis: An Introduction*, ed. M. Berthold and D. J. Hand, 269 – 298. Berlin: Springer.

Bottou, L., and V. Vapnik. 1992. "Local Learning Algorithms." *Neural Computation* 4: 888 – 900.

Broomhead, D. S., and D. Lowe. 1988. "Multivariable Functional Interpolation and Adaptive Networks." *Complex Systems* 2: 321 – 355.

Carpenter, G. A., and S. Grossberg. 1988. "The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network." *IEEE Computer* 21(3): 77 – 88.

302

Cherkassky, V., and F. Mulier. 1998. *Learning from Data: Concepts, Theory, and Methods.* New York: Wiley.

DeSieno, D. 1988. "Adding a Conscience Mechanism to Competitive Learning." In *IEEE International Conference on Neural Networks*, 117 – 124. Piscataway, NJ: IEEE Press.

Feldman, J. A., and D. H. Ballard. 1982. "Connectionist Models and their Properties." *Cognitive Science* 6: 205 – 254.

Fritzke, B. 1995. "Growing Cell Structures: A Self Organizing Network for Unsupervised and Supervised Training." *Neural Networks* 7: 1441 – 1460.

Grossberg, S. 1980. "How does the Brain Build a Cognitive Code?" *Psychological Review* 87: 1 – 51.

Hertz, J., A. Krogh, and R. G. Palmer. 1991. *Introduction to the Theory of Neural Computation.* Reading, MA: Addison Wesley.

Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton. 1991. "Adaptive Mixtures of Local Experts." *Neural Computation* 3: 79 – 87.

Jordan, M. I., and R. A. Jacobs. 1994. "Hierarchical Mixtures of Experts and the EM Algorithm." *Neural Computation* 6: 181 – 214.

Kohonen, T. 1990. "The Self-Organizing Map." *Proceedings of the IEEE* 78: 1464 – 1480.

Kohonen, T. 1995. *Self-Organizing Maps.* Berlin: Springer.

Lee, Y. 1991. "Handwritten Digit Recognition Using *k*-Nearest Neighbor, Radial Basis Function, and Backpropagation Neural Networks." *Neural Computation* 3: 440 – 449.

Mao, J., and A. K. Jain. 1995. "Artificial Neural Networks for Feature Extraction and Multivariate Data Projection." *IEEE Transactions on Neural Networks* 6: 296 – 317.

Moody, J., and C. Darken. 1989. "Fast Learning in Networks of Locally-Tuned Processing Units." *Neural Computation* 1: 281 – 294.

Oja, E. 1982. "A Simplified Neuron Model as a Principal Component Analyzer." *Journal of Mathematical Biology* 15: 267 – 273.

Omohundro, S. M. 1987. "Efficient Algorithms with Neural Network Behavior." *Complex Systems* 1: 273 – 347.

Platt, J. 1991. "A Resource Allocating Network for Function Interpolation." *Neural Computation* 3: 213 – 225.

Specht, D. F. 1991. "A General Regression Neural Network." *IEEE Transactions on Neural Networks* 2: 568 – 576.

Tresp, V., J. Hollatz, and S. Ahmad. 1997. "Representing Probabilistic Rules with Networks of Gaussian Basis Functions." *Machine Learning* 27: 173 – 200.

第13章 隐马尔可夫模型

我们放松样本实例相互独立的假设，并引入马尔可夫模型，将输入序列建模为由一个参数化随机过程所产生的序列。我们讨论如何完成这一建模以及从样本序列学习模型参数的算法。

13.1 引言

迄今为止，我们一直假设样本中的实例是独立同分布的。这样做的好处是样本的似然可简化为各个实例的似然之积。然而，这一假设在相继实例相互依赖的应用中并不成立。例如，在一个单词中相继的字母是相互依赖的；在英文中，“h”非常可能跟随在“t”而非“x”之后。这类存在观测序列(例如，单词中的字母，DNA序列中的基对)的过程并不能用简单的概率分布进行建模。一个类似的例子是语言识别，其中语音片段由称为音素的语音基元组成；只有某些音素序列是合法的，即该语言的单词。在更高的层次上，以某种序列书写或读出单词，形成由该语言的语法和语义规则定义的语句。

一个序列可用一个参数化的随机过程(parametric random process)来刻画。本章，我们讨论如何完成这种建模以及如何从样本序列的训练集中学习模型的参数。

13.2 离散马尔可夫过程

考虑一个系统，其在任意时刻处于N个离散状态中的一个：S_1，S_2，…，S_N。在时刻t的状态记作q_t，$t=1$，2，…。例如，$q_t=S_i$表示在时刻t系统处于状态S_i。尽管我们用“时刻”好像这应该是一个时间序列，但是这种方法对任意序列，无论是时间、空间、DNA串上位置等等，都是有效的。

系统在有规律的、间隔的离散时刻，根据以前的状态值，以给定的概率转移到一个状态：

$$P(q_{t+1}=S_j \mid q_t=S_i, q_{t-1}=S_k, \cdots)$$

对于一阶马尔可夫模型(Markov model)的特例，系统在时刻$t+1$的状态仅仅依赖于在时刻t的状态，而与之前的状态无关：

$$P(q_{t+1}=S_j \mid q_t=S_i, q_{t-1}=S_k, \cdots)=P(q_{t+1}=S_j \mid q_t=S_i) \tag{13.1}$$

这相当于说，给定当前的状态，未来的系统状态独立于过去的状态。这恰是谚语“今天是你余生的第一天”的数学表达版本。

我们进一步简化模型，假定转移概率(transition probability)是独立于时间的：

$$a_{ij} \equiv P(q_{t+1}=S_j \mid q_t=S_i) \tag{13.2}$$

满足

$$a_{ij} \geqslant 0 \text{ 并且 } \sum_{j=1}^{N} a_{ij}=1 \tag{13.3}$$

因此，从状态 S_i 到状态 S_j 的状态转移总是具有相同的概率，无论这个转移在观测序列中的何时或何地发生。$\mathbf{A}=[a_{ij}]$是一个 $N\times N$ 的矩阵，其每行之和均为 1。

这可看作是一个随机自动机(stochastic automation)(见图 13-1)。从每个状态 S_i，系统以概率 a_{ij}转移到状态 S_j，并且这一概率在任何时刻 t 均相同。唯一的特例是第一个状态。我们定义初始概率(initial probability)π_i，表示序列的第一个状态是 S_i 的概率：

$$\pi_i \equiv P(q_1 = S_i) \tag{13.4}$$

306

满足

$$\sum_{i=1}^{N} \pi_i = 1 \tag{13.5}$$

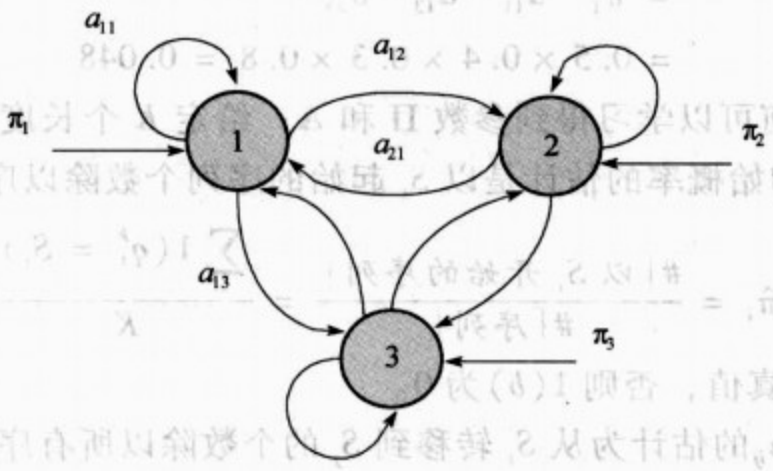


图 13-1 具有三个状态的马尔可夫模型作为随机自动机的实例。π_i 是系统始于状态 S_i 的概率，a_{ij}是系统从状态 S_i 转移到状态 S_j 的概率

$\Pi=[\pi_i]$是一个具有 N 个元素的向量，元素和为 1。

在一个可观测马尔可夫模型(observable Markov model)中，状态是可观测的。在任意时刻 t，我们知道 q_t，并且随着系统从一个状态转移到另一个状态，我们得到一个观测序列，即为状态序列。该过程的输出是一个状态集，在每个时间点上每个状态对应一个物理可观测事件。

我们有一个观测序列 O，它是状态序列 $O=Q=\{q_1q_2\cdots q_T\}$，其概率为

$$P(O=Q\mid \mathrm{A},\Pi)=P(q_1)\prod_{t=2}^{T}P(q_t\mid q_{t-1})=\pi q_1 a_{q_1q_2}\cdots a_{q_{T-1}q_T} \tag{13.6}$$

π_{q_1}是首状态为 q_1 的概率，$a_{q_1q_2}$是从 q_1 到 q_2 的概率，以此类推。我们将这些概率相乘，得到产生整个序列发生的概率。

我们现在看一个具体例子(Rabiner 和 Juang 1986)：假定我们有 N 个容器，每个容器中仅有一种颜色的球。例如，有一个装红色球的容器，一个装蓝色球的容器……。某人一个接一个地从各个容器中取球，并将它们的颜色展示给我们。以 q_t 表示在时刻 t 所取球的颜色。我们假定有三个状态：

307

$$S_1:\text{红},S_2:\text{蓝},S_3:\text{绿}$$

并有初始概率：

$$\Pi=[0.5,0.2,0.3]^T$$

a_{ij}是从容器 i 中取一个颜色 i 的球之后，从容器 j 中取(一个颜色 j 的)球的概率。例如，

转移矩阵为：

$$\mathbf{A}=\begin{bmatrix}0.4 & 0.3 & 0.3\\ 0.2 & 0.6 & 0.2\\ 0.1 & 0.1 & 0.8\end{bmatrix}$$

给定 Π 和 $\mathbf{A}$，很容易产生 K 个长度为 T 的随机序列。我们来看如何计算一个序列的概率：假定前四个球是“红，红，绿，绿”。这对应观测序列 $O=\{S_1, S_1, S_3, S_3\}$。其概率为

$$\begin{aligned}P(O\mid \mathbf{A},\Pi) &= P(S_1)\cdot P(S_1\mid S_1)\cdot P(S_3\mid S_1)\cdot P(S_3\mid S_3)\\ &= \pi_1\cdot a_{11}\cdot a_{13}\cdot a_{33}\\ &= 0.5\times 0.4\times 0.3\times 0.8 = 0.048\end{aligned} \tag{13.7}$$

现在，我们来看如何可以学习得到参数 Π 和 $\mathbf{A}$：给定 K 个长度为 T 的序列，q_t^k 表示序列 k 在 t 时刻的状态，初始概率的估计是以 S_i 起始的序列个数除以序列总数：

$$\hat{\pi}_i = \frac{\#\{\text{以 } S_i \text{ 开始的序列}\}}{\#\{\text{序列}\}} = \frac{\sum_k 1(q_1^k = S_i)}{K} \tag{13.8}$$

其中 $1(b)$ 为 1 如果 b 取真值，否则 $1(b)$ 为 0。

至于转移概率，对 a_{ij} 的估计为从 S_i 转移到 S_j 的个数除以所有序列中从 S_i 转移的总数：

$$\hat{a}_{ij} = \frac{\#\{\text{从 } S_i \text{ 到 } S_j \text{ 的转移}\}}{\#\{\text{从 } S_i \text{ 的转移}\}} = \frac{\sum_k \sum_{t=1}^{T-1} 1(q_t^k = S_i \text{ and } q_{t+1}^k = S_j)}{\sum_k \sum_{t=1}^{T-1} 1(q_t^k = S_i)} \tag{13.9}$$

308 $\hat{a}_{12}$ 是一个蓝色球跟在一个红色球之后的次数除以所有序列中红色球的总数。

13.3 隐马尔可夫模型

在隐马尔可夫模型(hidden Markov model，HMM)中，系统状态是不可观测的，但是到达一个状态时，可以记录一个观测，这个观测是该状态的一个概率函数。我们假定每个状态的一个离散观测取自集合 $\{v_1, v_2, \cdots, v_M\}$：

$$b_j(m) \equiv P(O_t = v_m \mid q_t = S_j) \tag{13.10}$$

$b_j(m)$ 是系统处于状态 S_j 时，我们观测到 v_m($m=1, \cdots, M$)的观测概率(observation probability)或发射概率(emission probability)。我们再次假定齐次模型，其中发射概率不依赖于时间 t。观测到的一系列 v_m 便形成了观测序列 O。状态序列 Q 是不可观测的，这正是称之为“隐”模型的缘由，但是状态序列可以通过观测序列推断。注意，通常许多不同的状态序列 Q 可以产生相同的观测序列 O，但是以不同的概率产生；正如，给定服从正态分布的一个独立同分布(iid)的样本，有无限多对可能的(μ，σ)值，我们感兴趣的是能以最大似然产生这个样本的那对(μ，σ)。

还需要注意的是，在隐马尔可夫模型中，随机性源自两个方面：除了从一个状态转移到另一状态是随机的之外，系统在一个状态中产生的观测也是随机的。

再次回到我们的例子：隐马尔可夫模型对应的容器-球实例中，每个容器包含不同颜色

的球。以 $b_j(m)$ 表示从容器 j 取出一个 m 颜色球的概率。我们再次得到一个球颜色的观测序列，但这次并不知道球取自哪个容器的序列。因此好像容器置于一个布帘之后，一个人随机地从一个容器中取一个球，而展示给我们的仅仅是球而不展示从哪个容器中取球。球展示后被放回容器以保持发射概率不变。球的颜色号码可能不同于容器号码。例如，我们假定有三个容器，而观测序列为：

$$O = \{\text{红},\text{红},\text{绿},\text{蓝},\text{黄}\}$$

在前面的情况下，知道观测(球的颜色)，我们可以确切知道系统状态(容器)，因为对不同的颜色的球有不同的容器，而且每个容器只含有一种颜色的球。可观测马尔可夫模型是隐马尔可夫模型的一个特例，其中 $M=N$，并且如果 $j=m$，$b_j(m)$ 为 1，否则 $b_j(m)$ 为 0。但是在隐马尔可夫模型中，一个球可能取自任意容器。在这种情况下，对于相同的观测序列 O，可能存在多个可能的状态序列 Q 产生 O(见图 13-2)。 309

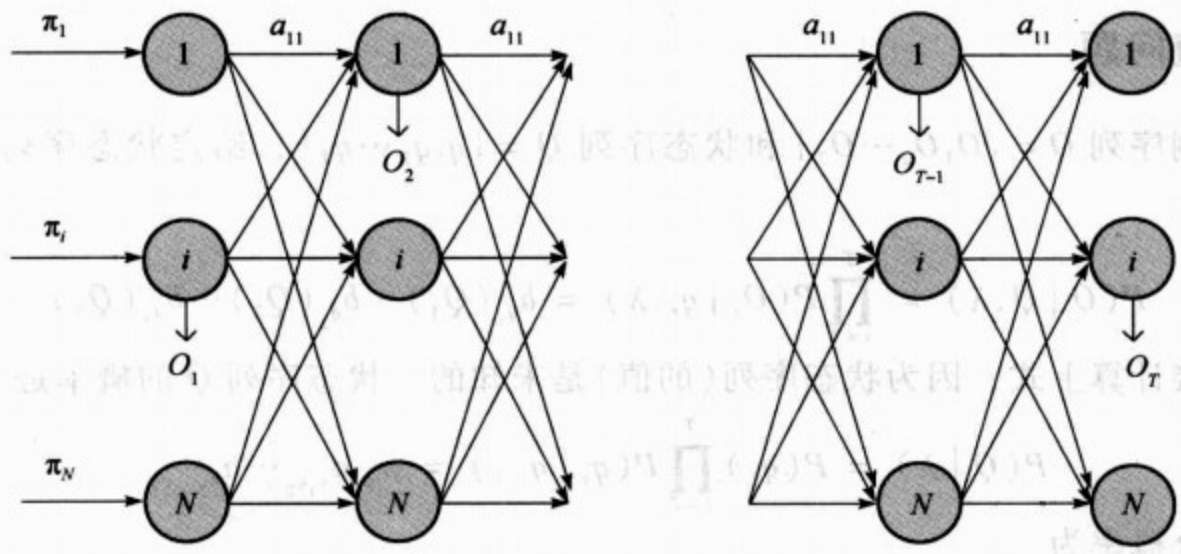


图 13-2 一个 HMM 按时间展开为格，展示了所有可能的轨道。以粗线所示的一条路径是产生观测序列的真正(未知)状态轨迹

对上述进行总结和形式化，一个 HMM 具有以下元素：

1. N：模型状态个数

$$S = \{S_1, S_2, \cdots, S_N\}$$

2. M：以字母序排列的不同观测符号的个数

$$V = \{v_1, v_2, \cdots, v_M\}$$

3. 状态转移概率：

$$\mathbf{A} = [a_{ij}], \qquad \text{其中 } a_{ij} \equiv P(q_{t+1} = S_j \mid q_t = S_i)$$

4. 观测概率：

$$\mathbf{B} = [b_j(m)], \qquad \text{其中 } b_j(m) \equiv P(O_t = v_m \mid q_t = S_j)$$ 310

5. 初始状态概率：

$$\Pi = [\pi_i], \qquad \text{其中 } \pi_i \equiv P(q_1 = S_i)$$

N 和 M 隐含地定义在其他参数中，因此 $\lambda = (\mathbf{A}, \mathbf{B}, \Pi)$ 被取作一个 HMM 的参数集。给定 λ，模型可用于产生任意长度的任意个数观测序列，但是我们通常感兴趣的是另一方向，即通过一个由若干观测序列组成的训练集来估计模型的参数。

13.4 HMM的三个基本问题

给定一定数量的观测序列，我们对下面三个问题感兴趣：

1. 给定一个模型 λ，我们希望估计任意给定观测序列 $O=\{O_1O_2\cdots O_T\}$ 的概率，即 $P(O\mid\lambda)$。

2. 给定一个模型 λ 及一个观测序列 O，我们希望找出状态序列 $Q=\{q_1q_2\cdots q_T\}$，其具有产生 O 的最大概率，即我们要找到最大化 $P(Q\mid O,\lambda)$ 的 Q^*。

3. 给定观测序列组成的训练集 $\chi=\{O^k\}_k$，我们希望学习得到产生 χ 的概率最大化的模型，即我们要找到最大化 $P(\chi\mid\lambda)$ 的 λ^*。

让我们逐一来看这些问题的解决办法，每个办法又用于解决下一个问题，直到我们得以计算出 λ，或者是从数据中学到一个模型。

13.5 估值问题

给定观测序列 $O=\{O_1O_2\cdots O_T\}$ 和状态序列 $Q=\{q_1q_2\cdots q_T\}$，给定状态序列 Q 观测到 O 的概率为

311

$$P(O\mid Q,\lambda)=\prod_{t=1}^{T}P(O_t\mid q_t,\lambda)=b_{q_1}(Q_1)\cdot b_{q_2}(Q_2)\cdots b_{q_T}(Q_T) \tag{13.11}$$

我们无法直接计算上式，因为状态序列(的值)是未知的。状态序列 Q 的概率是

$$P(Q\mid\lambda)=P(q_1)\prod_{t=2}^{T}P(q_t\mid q_{t-1})=\pi_{q_1}a_{q_1q_2}\cdots a_{q_{T-1}q_T} \tag{13.12}$$

因而联合概率为

$$\begin{aligned}P(O,Q\mid\lambda)&=P(q_1)\prod_{t=2}^{T}P(q_t\mid q_{t-1})\prod_{t=1}^{T}P(O_t\mid q_t)\\&=\pi_{q_1}b_{q_1}(O_1)a_{q_1q_2}b_{q_2}(O_2)\cdots a_{q_{T-1}q_T}b_{q_T}(O_T)\end{aligned} \tag{13.13}$$

我们可以通过边缘化这一联合概率，即通过在所有可能的 Q 上求和，来计算 $P(O\mid\lambda)$：

$$P(O\mid\lambda)=\sum_{\text{所有可能的}Q}P(O,Q\mid\lambda)$$

但是，该方法是不现实的，因为如果假定所有的概率都是非零的，则有 N^T 个可能的 Q。幸运的是，存在计算 $P(O\mid\lambda)$ 的有效方法，称之为正反向过程(forward-backward procedure)。其基于的思想是将观测序列分为两个部分：第一部分始于时刻1到时刻 t，第二部分始于时刻 $t+1$ 直到 T。

给定模型 λ，我们定义正向变量(forward variable) $\alpha_t(i)$ 为到时刻 t 并且在时刻 t 的状态为 S_i 时，观测到部分序列 $\{O_1\cdots O_t\}$ 的概率：

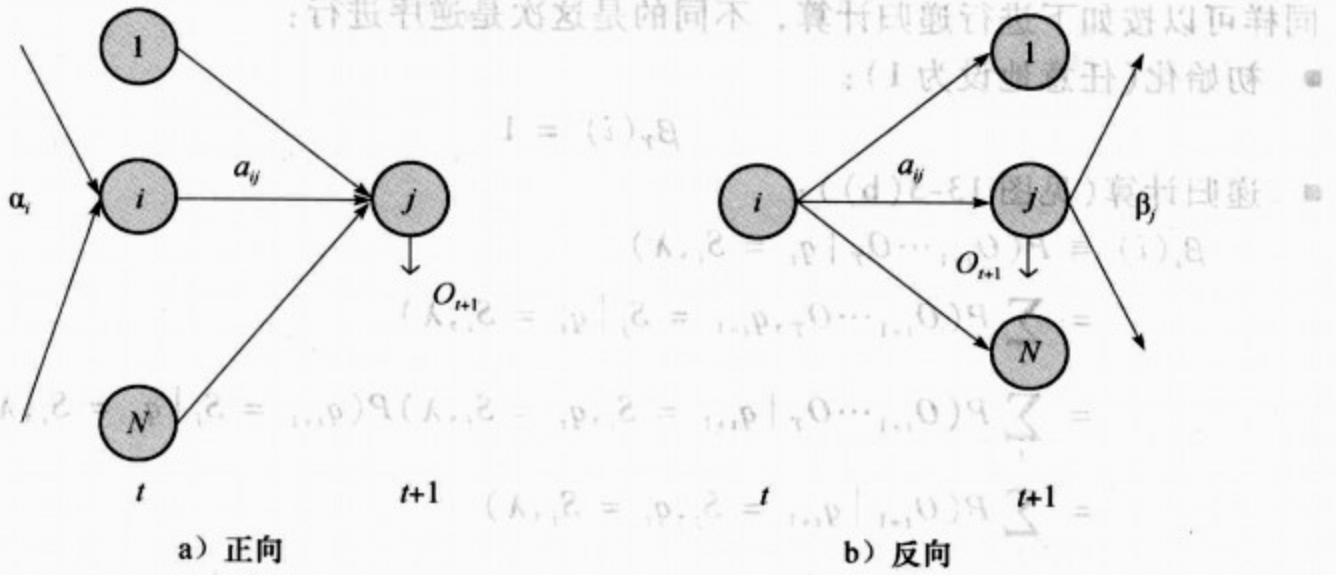
$$\alpha_t(i)\equiv P(O_1\cdots O_t,q_t=S_i\mid\lambda) \tag{13.14}$$

这种方法的优点在于可通过结果累积而递归地计算上式：

- 初始化：

$$\begin{aligned}\alpha_1(i)&\equiv P(O_1,q_1=S_i\mid\lambda)\\&=P(O_1\mid q_1=S_i,\lambda)P(q_1=S_i\mid\lambda)\\&=\pi_ib_i(O_1)\end{aligned} \tag{13.15}$$

■ 递归(见图 13-3(a)):



a) 正向　　b) 反向

图 13-3　正反向过程 a)$\alpha_t(j)$的计算；b)$\beta_t(i)$的计算

$$
\begin{aligned}
\alpha_{t+1}(j) &\equiv P(O_1\cdots O_{t+1}, q_{t+1} = S_j \mid \lambda)\\
&= P(O_1\cdots O_{t+1} \mid q_{t+1} = S_j, \lambda)P(q_{t+1} = S_j \mid \lambda)\\
&= P(O_1\cdots O_t \mid q_{t+1} = S_j, \lambda)P(O_{t+1} \mid q_{t+1} = S_j, \lambda)P(q_{t+1} = S_j \mid \lambda)\\
&= P(O_1\cdots O_t, q_{t+1} = S_j \mid \lambda)P(O_{t+1} \mid q_{t+1} = S_j, \lambda)\\
&= P(O_{t+1} \mid q_{t+1} = S_j, \lambda)\sum_i P(O_1\cdots O_t, q_t = S_i, q_{t+1} = S_j, \lambda)\\
&= P(O_{t+1} \mid q_{t+1} = S_j, \lambda)\sum_i P(O_1\cdots O_t, q_{t+1} = S_j \mid q_t = S_i, \lambda)P(q_t = S_i \mid \lambda)\\
&= P(O_{t+1} \mid q_{t+1} = S_j, \lambda)\sum_i P(O_1\cdots O_t \mid q_t = S_i, \lambda)P(q_{t+1} = S_j \mid q_t = S_i, \lambda)P(q_t = S_i \mid \lambda)\\
&= P(O_{t+1} \mid q_{t+1} = S_j, \lambda)\sum_i P(O_1\cdots O_t, q_t = S_i \mid \lambda)P(q_{t+1} = S_j \mid q_t = S_i, \lambda)\\
&= \Big[\sum_{i=1}^{N}\alpha_t(i)a_{ij}\Big]b_j(O_{t+1}) \qquad (13.16)
\end{aligned}
$$

$\alpha_t(i)$表示了止于状态 S_i 的前 t 个观测的概率。通过将其乘以概率 a_{ij}得到转移到状态 S_j 的概率，但是因为有 N 个可能的前驱状态，我们需要对所有可能的前驱状态 S_i 求和。$b_j(O_{t+1})$则是产生第$(t+1)$个观测且在第 $t+1$ 时刻处于状态 S_j 的概率。

312
t
313

当我们计算正向变量时，容易计算观测序列的概率：

$$
\begin{aligned}
P(O \mid \lambda) &= \sum_{i=1}^{N} P(O, q_T = S_i \mid \lambda)\\
&= \sum_{i=1}^{N} \alpha_T(i) \qquad (13.17)
\end{aligned}
$$

$\alpha_T(i)$是产生整个观测序列并终止于状态 S_j 的概率。我们需要对所有可能的终止状态进行求和。

计算 $\alpha_t(i)$的复杂度为$\mathcal{O}(N^2T)$，并且这在合理的时间内解决我们第一个估值问题。虽然现在我们并不需要，但是我们类似地定义反向变量(backward variable)$\beta_t(i)$，作为在时刻 t 处于状态 S_i 并且观测到部分序列 $O_{t+1}\cdots O_T$ 的概率：

$$\beta_t(i) \equiv P(O_{t+1}\cdots O_T \mid q_t = S_i, \lambda) \tag{13.18}$$

同样可以按如下进行递归计算，不同的是这次是逆序进行：

- 初始化(任意地设为1)：

$$\beta_T(i) = 1$$

- 递归计算(见图13-3(b))：

$$\begin{aligned}
\beta_t(i) &\equiv P(O_{t+1}\cdots O_T \mid q_t = S_i, \lambda) \\
&= \sum_j P(O_{t+1}\cdots O_T, q_{t+1} = S_j \mid q_t = S_i, \lambda) \\
&= \sum_j P(O_{t+1}\cdots O_T \mid q_{t+1} = S_j, q_t = S_i, \lambda) P(q_{t+1} = S_j \mid q_t = S_i, \lambda) \\
&= \sum_j P(O_{t+1} \mid q_{t+1} = S_j, q_t = S_i, \lambda) \\
&\quad P(O_{t+2}\cdots O_T \mid q_{t+1} = S_j, q_t = S_i, \lambda) P(q_{t+1} = S_j \mid q_t = S_i, \lambda) \\
&= \sum_j P(O_{t+1} \mid q_{t+1} = S_j, \lambda) \\
&\quad P(O_{t+2}\cdots O_T \mid q_{t+1} = S_j, \lambda) P(q_{t+1} = S_j \mid q_t = S_i, \lambda) \\
&= \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)
\end{aligned} \tag{13.19}$$

314

当处于状态 S_i 时，有 N 种可能的下一状态 S_j，每个的概率为 a_{ij}。在该状态上，我们产生第$(t+1)$个观测，而 $\beta_{t+1}(j)$表示了时刻 $t+1$ 之后的所有观测的概率，以此类推。

对于实现需要引起注意的是：α_t 和 β_t 的计算都是通过多个小概率相乘实现的，而当序列很长的时候可能发生下溢。为了避免下溢，我们在每一步通过将 $\alpha_t(i)$除以 $c_t = \sum_j \alpha_t(j)$ 对其进行规范化。同样将 $\beta_t(i)$除以相同的 c_t 对其进行规范化($\beta_t(i)$之和不为1)。

13.6 寻找状态序列

我们现在考虑第二个问题，即给定模型 λ，寻找以最高的概率产生观测序列 $O = \{O_1 O_2 \cdots O_T\}$ 的状态序列 $Q = \{q_1 q_2 \cdots q_T\}$。

定义 $\gamma_t(i)$为给定 O 和 λ，在时刻 t 处于状态 S_i 的概率，它可以计算如下：

$$\gamma_t(i) \equiv P(q_t = S_i \mid O, \lambda) \tag{13.20}$$

$$\begin{aligned}
&= \frac{P(O \mid q_t = S_i, \lambda) P(q_t = S_i \mid \lambda)}{P(O \mid \lambda)} \\
&= \frac{P(O_1\cdots O_t \mid q_t = S_i, \lambda) P(O_{t+1}\cdots O_T \mid q_t = S_i, \lambda) P(q_t = S_i \mid \lambda)}{\sum_{j=1}^{N} P(O, q_t = S_j \mid \lambda)} \\
&= \frac{P(O_1\cdots O_t, q_t = S_i \mid \lambda) P(O_{t+1}\cdots O_T \mid q_t = S_i, \lambda)}{\sum_{j=1}^{N} P(O \mid q_t = S_j, \lambda) P(q_t = S_j \mid \lambda)} \\
&= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}
\end{aligned} \tag{13.21}$$

这里我们看到 $\alpha_t(i)$ 和 $\beta_t(i)$ 是如何很好的在它们间分割序列：正向变量 $\alpha_t(i)$ 表示了直到时刻 t 的开始部分序列并止于状态 S_i 的概率，而反向变量 $\beta_t(i)$ 从那里开始并表示了直到时刻 T 的结尾部分的概率。

分子 $\alpha_t(i)\beta_t(i)$ 表示了在时刻 t、系统处于状态 S_i 的整个序列的概率。我们需要将其除 315
以所有在时刻 t 可能转移到的中间状态对其进行正规化，并保证 $\sum_i \gamma_t(i) = 1$。

为找到(目标)状态序列，可以在每一步 t 选择具有最高概率的状态：

$$q_t^* = \arg\max_i \gamma_t(i) \tag{13.22}$$

但是这有可能在时刻 t 和时刻 $t+1$ 选择 S_i 和 S_j 作为最合适的状态，即使这时有 $a_{ij}=0$。为了找到单个最好的状态序列(路径)，我们使用基于动态规划的 Viterbi 算法(Viterbi algorithm)，将这样的转移概率考虑在内。

给定状态序列 $Q = q_1 q_2 \cdots q_T$ 和观测序列 $O = O_1 \cdots O_T$，定义 $\delta_t(i)$ 为在时刻 t 代表前 t 个观测并止于状态 S_i 具有最高概率路径的概率：

$$\delta_t(i) \equiv \max_{q_1 q_2 \cdots q_{t-1}} p(q_1 q_2 \cdots q_{t-1}, q_t = S_i, O_1 \cdots O_t \mid \lambda) \tag{13.23}$$

从而我们可以递归地计算 $\delta_{t+1}(i)$，而最优路径可以从时刻 T 开始，在每个时刻选择最合适的状态来反向读取。算法如下：

1. 初始化：

$$\delta_1(i) = \pi_i b_i(O_1)$$
$$\psi_1(i) = 0$$

2. 递归：

$$\delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} \cdot b_j(O_t)$$
$$\psi_t(j) = \arg\max_i \delta_{t-1}(i) a_{ij}$$

3. 终止：

$$p^* = \max_i \delta_T(i)$$
$$q_T^* = \arg\max_i \delta_T(i)$$

4. 路径(状态序列)回溯：

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \cdots, 1 \quad 316$$

使用图 13-2 的网格结构，$\psi_t(j)$ 跟踪了在时刻 $t-1$ 最大化 $\delta_t(j)$ 的状态，也就是说，最佳的前驱状态。在正向阶段 Viterbi 算法有相同的复杂度，其中我们在每一步用取最大值替代求和。

13.7 学习模型参数

我们现在继续第三个问题，从数据中学习 HMM。使用最大似然方法，我们要计算最大化训练序列样本 $\chi = \{O^k\}_{k=1}^K$ 的似然的 λ^*，即计算最大化 $P(\chi \mid \lambda)$ 的 λ^*。我们从定义便于稍后讨论的新变量开始。

定义 $\xi_t(i, j)$ 为给定全部观测 O 和 λ，在时刻 t 处于状态 S_i 和在时刻 $t+1$ 处于状态 S_j 的概率：

$$\xi_t(i,j) \equiv P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) \tag{13.24}$$

上式可计算为(见图13-4)：

$$\begin{aligned}
\xi_t(i,j) &\equiv P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) \\
&= \frac{P(O \mid q_t = S_i, q_{t+1} = S_j, \lambda) P(q_t = S_i, q_{t+1} = S_j \mid \lambda)}{P(O \mid \lambda)} \\
&= \frac{P(O \mid q_t = S_i, q_{t+1} = S_j, \lambda) P(q_{t+1} = S_j \mid q_t = S_i, \lambda) P(q_t = S_i \mid \lambda)}{P(O \mid \lambda)} \\
&= \left(\frac{1}{P(O \mid \lambda)}\right) P(O_1 \cdots O_t \mid q_t = S_i, \lambda) P(O_{t+1} \mid q_{t+1} = S_j, \lambda) \\
&\quad P(O_{t+2} \cdots O_T \mid q_{t+1} = S_j, \lambda) a_{ij} P(q_t = S_i \mid \lambda) \\
&= \left(\frac{1}{P(O \mid \lambda)}\right) P(O_1 \cdots O_t, q_t = S_i \mid \lambda) P(O_{t+1} \mid q_{t+1} = S_j, \lambda) \\
&\quad P(O_{t+2} \cdots O_T \mid q_{t+1} = S_j, \lambda) a_{ij} \\
&= \frac{\alpha_t(i) b_j(O_{t+1}) \beta_{t+1}(j) a_{ij}}{\sum_k \sum_l P(q_t = S_k, q_{t+1} = S_l, O \mid \lambda)} \\
&= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_k \sum_l \alpha_t(k) a_{kl} b_l(O_{t+1}) \beta_{t+1}(l)}
\end{aligned} \tag{13.25}$$

$\alpha_t(i)$表示在时刻t的前t个观测并止于状态S_i的概率。以概率a_{ij}转移到态S_j，产生第$t+1$个观测，并在$t+1$时刻从S_j开始继续产生其余的观测序列。我们通过将$\xi_t(i, j)$除以所有在时刻t和时刻$t+1$可能处于的状态对其进行规范化。

如果需要，我们可以通过对所有可能的下一状态，在弧概率上边缘化来计算在t时刻系统处于状态S_i的概率：

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{13.26}$$

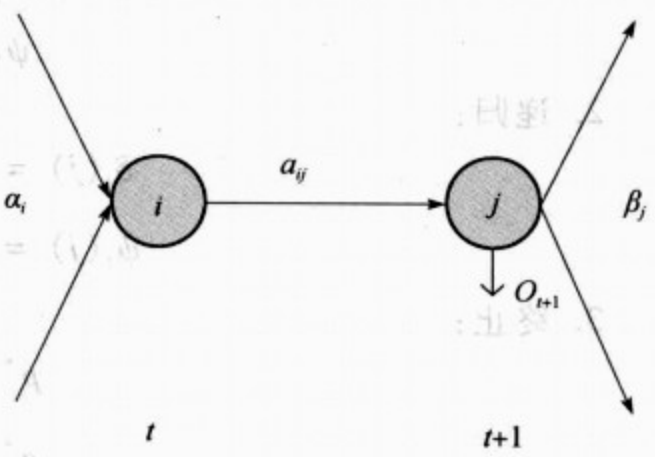


图13-4 计算弧概率$\xi_t(i, j)$

需要注意的是，如果马尔可夫模型不是隐的而是可观测的，则$\gamma_t(i)$和$\xi_t(i, j)$二者均为0/1。而当模型是隐马尔可夫模型时，我们通过软计数(soft count)这样的后验概率来估计它们。正如监督分类和非监督聚类之间的区别，其中类标号相应的为已知和未知。在使用EM算法的非监督聚类中(7.4节)，类标号未知，我们首先(在E-步骤中)估计它们，而后(在M-步骤中)使用这些估计计算参数。

类似地，这里我们使用Baum-Welch算法(Baum-Welch algorithm)，它是一种EM方法。在每次迭代中，首先在E-步，我们在给定当前$\lambda = (\mathbf{A}, \mathbf{B}, \Pi)$情况下计算$\xi_t(i, j)$和$\gamma_t(i)$的值，然后在M-步，我们在给定$\xi_t(i, j)$和$\gamma_t(i)$的情况下再计算$\lambda$。这两个步骤交替进行直到收敛，这是因为，$P(O \mid \lambda)$的值在这个过程中不会减小。

假设指示变量z_i^t为：

$$z_i^t = \begin{cases} 1 & \text{如果 } q_t = S_i \\ 0 & \text{否则} \end{cases} \tag{13.27}$$

并且

$$z_{ij}^t = \begin{cases} 1 & \text{如果 } q_t = S_i \text{ 并且 } q_{t+1} = S_j \\ 0 & \text{否则} \end{cases} \tag{13.28}$$

这些值在可观测马尔可夫模型情况下为0/1，而在HMM情况下为隐随机变量。在后一种情况下，我们在E-步对其进行估计：

$$E[z_i^t] = \gamma_t(i)$$
$$E[z_{ij}^t] = \xi_t(i,j) \tag{13.29}$$

在M-步，我们在这些估计值上计算参数。从 S_i 到 S_j 的转移的期望数为 $\sum_t \xi_t(i,j)$，而从 S_i 转移的总数为 $\sum_t \gamma_t(i)$。这两个数值的比值给出了任意时刻从状态 S_i 转移到 S_j 的概率：

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{13.30}$$

注意除了将实际的计数替换为估计的软计数外，上式和(13.9)式是一样的。

在状态 S_j 观测到 v_m 的概率为系统处于 S_j 状态时观测到 v_m 的期望次数除以系统处于 S_j 状态的总数：

$$\hat{b}_j(m) = \frac{\sum_{t=1}^{T} \gamma_t(j) 1(O_t = v_m)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{13.31}$$

当有多个观测序列 $\chi = \{O^k\}_{k=1}^K$ 时，我们假定它们是相互独立的：

$$P(\chi \mid \lambda) = \prod_{k=1}^{K} P(O^k \mid \lambda)$$

参数在全部序列的所有观测上取平均：

$$\hat{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T_K-1} \xi_t^k(i,j)}{\sum_{k=1}^{K} \sum_{t=1}^{T_K-1} \gamma_t^k(i)}$$

$$\hat{b}_j(m) = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T_K} \gamma_t^k(j) 1(O_t^k = v_m)}{\sum_{k=1}^{K} \sum_{t=1}^{T_K-1} \gamma_t^k(j)}$$

$$\hat{\pi}_i = \frac{\sum_{k=1}^{K} \gamma_1^k(i)}{K} \tag{13.32}$$

13.8 连续观测

在我们的讨论中，我们假定离散的观测服从多项分布：

$$P(O_t \mid q_t = S_j, \lambda) = \prod_{m=1}^{M} b_j(m)^{r_m^t} \tag{13.33}$$

其中

$$r_m^t = \begin{cases} 1 & \text{如果 } O_t = v_m \\ 0 & \text{否则} \end{cases} \tag{13.34}$$

如果输入是连续的，一个方法是将其离散化，然后使用这些离散值作为观测值。通常使用向量量化(7.3节)，目的在于将连续值转换为最接近的参考向量的离散值。例如，在语音识别中，一个单词发音被分割为小的语音片段，对应音节或部分音节；预处理后，这些片段通过向量量化被离散化，进而使用HMM将一个单词发音建模为一个离散化片段的序列。

我们还记得用于向量量化的k-均值是高斯混合模型的一个硬版本：

$$p(O_t \mid q_t = S_j, \lambda) = \sum_{l=1}^{L} P(\mathcal{G}_l) p(O_t \mid q_t = S_j, \mathcal{G}_l, \lambda) \tag{13.35}$$

其中

$$p(O_t \mid q_t = S_j, \mathcal{G}_l, \lambda) \sim \mathcal{N}(\mu_l, \Sigma_l) \tag{13.36}$$

并且观测保持连续性。在这种高斯混合情形下，可为分量参数(以合适的正则化来保持对参数个数进行检验)和混合比例推导出EM方程(Rabiner 1989)。

现在我们看一下观测为连续标量的情形，$O_t \in \mathfrak{R}$。最简单的方法是假定其服从正态分布：

$$p(O_t \mid q_t = S_j, \lambda) \sim \mathcal{N}(\mu_j, \sigma_j^2) \tag{13.37}$$

这意味在状态S_j，观测取自均值为μ_j，方差为σ_j^2的正态分布。这种情形下，M步的公式为：

320

$$\hat{\mu}_j = \frac{\sum_t \gamma_t(j) O_t}{\sum_t \gamma_t(j)} \tag{13.38}$$

$$\hat{\sigma}_j^2 = \frac{\sum_t \gamma_t(j)(O_t - \hat{\mu}_j)^2}{\sum_t \gamma_t(j)}$$

13.9 带输入的HMM

在某些应用中，除了观测序列O_t之外，我们还有一个输入序列x_t。我们可以将处于状态S_j的观测O_t以输入x_t为条件，并记作$P(O_t \mid q_t = S_j, x_t)$。当观测为连续标量的情况下，我们将(13.37)式替换为一个广义模型

$$p(O_t \mid q_t = S_j, x_t, \lambda) \sim \mathcal{N}(g_j(x^t \mid \theta_j), \sigma_j^2) \tag{13.39}$$

其中，比如假定线性模型，我们有

$$g_j(x^t \mid w_j, w_{j0}) = w_j x^t + w_{j0} \tag{13.40}$$

如果观测是离散的并服从多项分布，则我们得到一个将x_t作为输入并产生M选1输出

的分类器，否则我们可以产生后验类概率并保持观测的连续性。

类似地，状态转移概率也可以以输入为条件，即 $P(q_{t+1}=S_j, \mid q_t=S_i, x_t)$。它可通过一个选择将 $t+1$ 时刻的状态作为在 t 时刻的状态和输入的函数的分类器来实现。这就是马尔可夫混合专家模型(markov mixture of expert)(Meila 和 Jordan 1996)，是混合专家构架的一般化(见 12.8 节)，其中门网络跟踪其在前一时步所作的决策。此类构架也称为输入-输出 HMM(input-output HMM)(Bengio 和 Frasconi 1996)并具有模型非齐次的优点；在不同的时步使用不同的观测和转移概率。在每一个状态仍然使用由 θ_j 参数化的单一模型，但是根据所看到的输入产生不同的转移或观测概率。有可能输入并非单值，而是围绕时刻 t 的一个窗口，使得输入为一个向量；这样可以处理输入和观测序列具有不同长度的应用。

即使没有其他显式的输入序列，带输入的 HMM 也可以通过关于一些以前观测的预设函数来产生一个“输入”

$$x_t = f(O_{t-\tau}, \cdots, O_{t-1})$$

从而提供一个大小为 τ 的背景输入窗口。 321

13.10 HMM 中的模型选择

正如任意其他模型一样，需要调整 HMM 的复杂度，使其根据面对的数据的规模和性质平衡其复杂性。一种可能的方法是调整 HMM 的拓扑。在完全连接(遍历)的 HMM 中，从一个状态可转移到任意其他状态，使得 **A** 是一个 $N \times N$ 的全矩阵。在一些应用中，仅有某些转移是允许的，而不允许的转移有 $a_{ij}=0$。当下一个可能的状态较少，即 $N' < N$ 时，正反向扫描和 Viterbi 过程的复杂度为 $\mathcal{O}(NN'T)$，而并非 $\mathcal{O}(N^2T)$。

例如，在语音识别中使用*自左向右* HMM(left-to-right HMM)，其中系统状态按时间排序，随着时间的推进，状态下标增加或保持不变。这样的约束可用来对像语音一样其性质随时间变化的序列进行建模，并且当到达一个状态的时候，我们近似地知道其前的状态。有一个性质：系统绝不向具有更小下标的状态转移，即对于 $j<i$ 有 $a_{ij}=0$。而在状态下标上跨度很大的状态转移也不允许，即对于 $j>i+\tau$ 有 $a_{ij}=0$。图 13-5 给出了自左向右 HMM 的一个例子，其中 $\tau=2$，状态转移矩阵如下

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

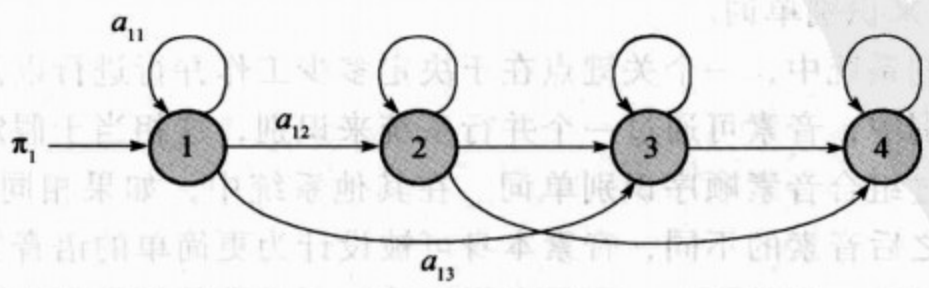


图 13-5 自左向右 HMM 的例子

决定 HMM 复杂度的另一因素是状态的个数 N。因为状态是隐藏的，因此其个数未知并且应在训练前选定。这需要使用先验信息对其进行决定并通过交叉确认，即通过检查确认序列的似然进行微调。

当用于分类时，可使用一组 HMM，每个对属于一类的序列进行建模。例如，在口语单词识别中，每个单词的样本训练一个不同的模型 λ_i。当对新的单词发音 O 进行分类时，所有不同的单词模型均被用来计算 $P(O \mid \lambda_i)$。然后使用贝叶斯规则得到后验概率

$$P(\lambda_i \mid O) = \frac{P(O \mid \lambda_i)P(\lambda_i)}{\sum_j P(O \mid \lambda_j)P(\lambda_j)} \tag{13.41}$$

其中 $P(\lambda_i)$ 是单词 i 的先验概率。该发音被指派到具有最高后验概率的单词。这是基于似然的方法，也是直接训练有判别力的 HMM，以便最大化后验概率。当存在同一单词的多个发音时，它们在该单词的 HMM 中被定义为并行路径。

在像语音这样的连续输入的情况下，困难之处在于将信号分割为较小的离散观测。通常使用取作基元部分的音素(phone)，并通过对其组合形成更长的序列(例如单词)。每个音素(通过向量量化)被并行地识别，然后用 HMM 将它们顺序组合。如果语音基元简单，则 HMM 会比较复杂，反之亦然。在连续语音识别中，单词并非一个接一个的以清晰间隔进行发音，这时可以采用多级别上的层次 HMM；一层用于组合音素以识别单词，另一层通过建立语言模型对单词组合以识别语句等。

近年来，神经网络/HMM 混合模型在语音识别领域比较流行(Morgan 和 Bourlard 1995)。在这样的模型中，一个多层感知器(第 11 章)用于捕捉时间局部而不是比较复杂和非线性的基元，如音素，而 HMM 用于学习时间结构。神经网络作为预处理器并将时间窗口中的原始观测转换成比向量量化的输出更容易建模的形式。

13.11 注释

322
~
323

HMM 是一项成熟的技术，并且存在基于 HMM 的商业语音识别系统在实际使用(Rabiner 和 Juang 1993；Jelinek 1997)。在 11.12 节，我们讨论了如何训练多层感知器用于序列识别。与延迟神经网络相比，HMM 的优点在于不用事先定义时间窗口，并且 HMM 的训练效果优于递归神经网络。HMM 可以应用于各种序列识别任务。HMM 在生物信息领域的应用在 Baldi 和 Brunak 1998 中有所介绍，在自然语音处理中的应用在 Manning 和 Schütz 1999 有所介绍。HMM 也用于在线手写符号识别，它与光学识别的不同之处在于书写者在触觉感知的书写板上书写，并且输入是一个关于笔尖在书写板上移动的 (x, y) 坐标序列，而非静态的图片。Bengio 等(1995)介绍了一种用于在线识别的混合系统，其中 MLP 识别单个字符，而 HMM 将字符组合起来识别单词。

在任意此类识别系统中，一个关键点在于决定多少工作并行进行以及将什么工作留做串行处理。在语音识别中，音素可通过一个并行系统来识别，这相当于假定所有的音素声音同时发出。然后，通过组合音素顺序识别单词。在其他系统中，如果相同的音素有多个版本，比如根据之前以及之后音素的不同，音素本身可被设计为更简单的语音发声的序列。并行工作是好的，但仅仅是在一定程度上；我们应当在并行和顺序处理之间找到理想的平衡。为了可以一键式接通任意人的电话，我们可能需要电话上有百万按键；作为替代，我们用十个按

键并以序列按键来拨通号码。

Bengio 1999 讨论了 HMM 的多种应用以及多种扩展，如有判别力的 HMM。HMM 可看作是贝叶斯网络(3.7 节)，HMM 上的推断及学习操作和贝叶斯网络中相应部分类似(Smyth、Heckerman 和 Jordan 1997)。最近提出了各种 HMM 的扩展，如因子 HMM，其中每一时间步骤，一定数量的状态集体生成观测；以及树结构 HMM，其中存在状态的一个层次关系。Ghahramani(2001)的介绍性论文对这些扩展及其训练的近似方法进行了讨论。 324

13.12 习题

1. 给定一个有三个状态 S_1，S_2，S_3 的可观测马尔可夫模型，其初始概率为：

$$\Pi = [0.5, 0.2, 0.3]^T$$

转移概率为

$$\mathbf{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

产生 100 个有 1000 个状态的序列。

2. 使用上题中产生的数据来估计 Π 和 $\mathbf{A}$，并和产生这些数据的参数进行比较。
3. 形式化一个二阶马尔可夫模型。其参数是什么？如何对一个给定的状态序列计算其概率？对于一个可观测模型如何学习其参数？
4. 证明任意二阶(或更高阶)马尔可夫模型可转化为一个一阶马尔可夫模型。
5. 一些研究者将马尔可夫模型定义为当穿越一条弧(边)的时候产生一个观测，而非到达一个状态。这种模型的能力是否比我们讨论的模型更强？
6. 从一个你选择的 HMM 产生训练和确认序列。在相同的训练集上通过变化隐藏状态的个数来训练多个不同的 HMM 并计算相应的确认似然。观察确认似然如何随着状态个数的增加而变化。

13.13 参考文献

Baldi, P., and S. Brunak. 1998. *Bioinformatics: The Machine Learning Approach*. Cambridge, MA: The MIT Press.

Bengio, Y. 1999. "Markovian Models for Sequential Data." *Neural Computing Surveys* 2: 129 – 162.

Bengio, Y., and P. Frasconi. 1996. "Input-Output HMMs for Sequence Processing." *IEEE Transactions on Neural Networks* 7: 1231 – 1249.

Bengio, Y., Y. Le Cun, C. Nohl, and C. Burges. 1995. "LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition." *Neural Computation* 7: 1289 – 1303.

Ghahramani, Z. 2001. "An Introduction to Hidden Markov Models and Bayesian Networks." *International Journal of Pattern Recognition and Artificial Intelligence* 15: 9 – 42. 325

Jelinek, F. 1997. *Statistical Methods for Speech Recognition*. Cambridge, MA: The MIT Press.

Manning, C. D., and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: The MIT Press.

Meila, M., and M. I. Jordan. 1996. "Learning Fine Motion by Markov Mixtures of Experts." In *Advances in Neural*

Information Processing Systems 8, ed. D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, 1003 – 1009. Cambridge, MA: The MIT Press.

Morgan, N., and H. Bourlard. 1995. "Continuous Speech Recognition: An Introduction to the Hybrid HMM/Connectionist Approach." *IEEE Signal Processing Magazine* 12: 25 – 42.

Smyth, P., D. Heckerman, and M. I. Jordan. 1997. "Probabilistic Independence Networks for Hidden Markov Probability Models." *Neural Computation* 9: 227 – 269.

Rabiner, L. R. 1989. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE* 77: 257 – 286.

Rabiner, L. R., and B. H. Juang. 1986. "An Introduction to Hidden Markov Models." *IEEE Acoustics, Speech, and Signal Processing Magazine* 3: 4 – 16.

326 Rabiner, L. R., and B. H. Juang. 1993. *Fundamentals of Speech Recognition*. New York: Prentice Hall.

第 14 章　分类算法评估和比较

机器学习算法产生的分类器依赖于训练集，并需对分类器进行统计检验来(i)评估分类算法的期望误差率，以及(ii)比较两个分类算法的期望误差率，以判别哪个更好。本章，我们回顾假设检验并讨论评估和比较误差率的检验。

14.1　引言

在前面章节中，我们讨论了几种分类算法，并知道对于一个给定的应用，多种分类算法都是可行的。现在，我们关心的是以下两个问题：

1. 如何评估一个分类算法在给定问题上的期望误差率？即，已使用分类算法训练一个分类器，我们是否能够以足够的置信度说在之后的实际应用中，其期望误差率将小于，比如说，2%？

2. 给定两个分类算法，如何就给定的应用来判断一个算法的误差率比另一个低？进行比较的分类算法可能是不同的，例如，参数的与非参数的，抑或它们可能使用不同的超参数设置。例如，给定一个具有 4 个隐藏单元的多层感知器(第 11 章)和另一个具有 8 个隐藏单元的感知器，我们希望可以判断哪个具有更低的期望误差。或者在使用 k-最近邻分类器(第 8 章)时，我们希望找到最佳的 k 值。

我们不能只看训练集上的误差并据此来判定。根据定义，训练集上的误差率总会小于包含训练时未见过的实例的检验集上的误差率。类似地，训练误差不能用于比较两个算法。因为在训练集上，具有更多参数的较复杂模型几乎总是比简单模型的误差更小。

因此，正如我们反复讨论的，我们需要一个不同于训练集的确认集。而且即使是在确认集上，一轮运行也未必足够。其原因有二：首先，训练集和确认集可能较小并且可能包含异常实例，如噪声或离群点，可能会对我们产生误导。第二个原因在于学习算法有可能依赖于其他影响泛化的随机因素。例如，对于使用后向传播训练的一个多层感知器，由于梯度下降收敛于局部极小，初始权重会影响最终的权重，并且以完全相同的结构和训练集，以不同的初始权重开始训练有可能最终产生多种分类器，这些分类器在相同的确认集上有不同的误差率。因而我们需要多轮运行，以期在这些随机源上取平均值。如果我们只是训练和确认一次，则无法检验这些因素的影响；只有在学习方法的代价很高以至于只能训练和确认一次，这才是可以接受的。

我们在一个数据集上运行*分类算法*(classification algorithm)来产生一个分类器。如果我们只训练一次，我们只得到一个分类器和一个确认误差。为了平均(来自训练数据、初始权重等的)各种随机性，我们使用相同的算法来产生多个分类器。进而在多个确认集上检验这些分类器并记录确认误差的一个样本。(当然，所有训练和确认集均应取自同一应用)。我们对分类算法的评估基于这些*确认误差的分布*(distribution)。我们可使用这一分布来评估分类算法在应用问题上的*期望误差率*(expected error rate)，或者将其与其他分类算法的误差

率分布进行比较。

在讨论这一过程如何完成之前，需要重点强调以下几点：

1. 需要牢记的是：无论从我们的分析中获得何种结论，该结论限于所给定的数据集。
328 我们并不用独立于领域的方式来对分类算法进行比较，而是针对某一特定应用进行比较。一般而言，我们不对学习算法的期望误差率做任何讨论，也不将一个学习算法和另一个进行比较。我们所得到的任何结果只对这个特定的应用有效，而且仅在该应用可以由我们所使用的样本代表的意义上有效。而且无论如何，都没有诸如“最好的”学习算法之说。对任意学习算法，均有一个数据集使其非常准确，而另一个数据集使其非常差。当我们说一个分类算法好时，我们只是量化其归纳偏倚在多大程度上与数据的性质一致。这称之为*没有免费的午餐法则*(No free lunch theorem)。

2. 将给定数据划分为一定数量的训练集和确认集对仅仅是为了试验。一旦所有的试验完成，并且决定了最终方法或超参数，为了训练最终的分类器，我们可以使用先前用于训练或确认的所有已标记数据。

3. 由于我们还使用确认集进行试验，比如，选择两个分类算法中较好的一个，或决定何时停止学习，因此确认集实际上成为了我们所使用数据的一部分。在结束所有的试验之后，我们选定了某一特定的分类算法并且希望报告其期望误差率，为此我们应使用另外一个在训练最终系统过程中未曾使用过的检验集（test set）。该数据应当在之前的训练或确认过程中从未使用过，并且应足够大使得误差估计有意义。因此，给定一个数据集，我们应当留一部分数据作为检验集，而其余的数据用于训练和确认。通常，像我们在下一节中看到的那样，我们可以留三分之一的样本作为检验集，使用另外三分之二做交叉确认以产生多对训练/确认集。因而，给定特定学习算法和模型结构，训练集是用于参数优化；确认集是用于优化学习算法或模型结构的超参数；而一旦二者均被优化，检验集才在最后使用。例如，对一个 MLP(多层感知器)而言，训练集用于优化权重，确认集用于确定隐藏单元个数、训练多久、学习率等。一旦选择了最佳的 MLP 配置，其最终的误差率在检验集上计算。而
329 对 k-NN 而言，训练集作为查找表存放；我们在确认集上优化距离度量和 k 值，最后在检验集上进行检验。

4. 本章，我们就误差率对分类算法比较，但应牢记，在现实中，误差仅仅是影响决策的一个标准。一些其他标准是(Turney 2000)：

- 当使用损失函数，而非 0/1 损失对误差进行泛化时的风险(3.3 节)。
- 训练时间和空间复杂度。
- 检验时间和空间复杂度。
- 可解释性，即使用的方法是否允许提取可以由专家检查和确认的知识。
- 易于编程。

上述标准的相对重要程度依赖于应用。例如，如果在工厂中只进行一次训练，那么训练的时间和空间复杂度就不重要；如果在应用中对可适应性有要求，则训练的时间和空间复杂度就变得重要了。多数学习算法使用 0/1 损失并以误差最小化为唯一标准；最近，提出了这些算法的变种，即*代价敏感学习*(cost-sensitive learning)算法把其他标准也考虑在内。

14.2 交叉确认和再抽样方法

我们的第一个需求是从一个数据集$\mathcal{X}$中获得一定数目的训练集/确认集对。为此，如果样本$\mathcal{X}$足够大，我们可以随机地将其分为K个部分，然后将每一部分随机地分为两部分，一半用于训练，另一半用于确认。K通常为10或30。不幸的是，数据集从未有如此之大。因此我们应在小数据集上尽力而为。其方法是以不同分割来重复使用它；这称为交叉确认(cross-validation)。其潜在的问题是交叉确认使得错误比例是相互依赖的，因为这些不同集合共享了数据集。

因此，给定一个数据集$\mathcal{X}$，我们希望可以从该数据集产生K对训练/确认集$\{\mathcal{T}_i, \mathcal{V}_i\}_{i=1}^{K}$。我们要保持训练和确认集尽可能大，以保证误差估计的鲁棒性；同时，要保持不同集合间的交集尽可能小。我们还要确保当抽取数据子集的时候，类信息以正确比例被表示，从而使类的先验概率不受影响；这称为分层(stratification)。如果一个类在整个数据集中占有20%的实例，则在所有取自该数据集的抽样集中，该类也应约占20%的实例。 330

14.2.1 K-折交叉确认

在K-折交叉确认(K-fold cross-validation)中，数据集$\mathcal{X}$被随机的划分为K等份$\mathcal{X}_i$，$i=1, \cdots, K$。为了产生一对训练/确认集，我们将K份数据中的一份保留为确认集，其余$K-1$份合并为训练集。重复K次，每次保留K份中的另一份数据，可得到K对数据集：

$$\mathcal{V}_1 = \mathcal{X}_1 \quad \mathcal{T}_1 = \mathcal{X}_2 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K$$
$$\mathcal{V}_2 = \mathcal{X}_2 \quad \mathcal{T}_2 = \mathcal{X}_1 \cup \mathcal{X}_3 \cup \cdots \cup \mathcal{X}_K$$
$$\vdots$$
$$\mathcal{V}_K = \mathcal{X}_K \quad \mathcal{T}_K = \mathcal{X}_1 \cup \mathcal{X}_2 \cup \cdots \cup \mathcal{X}_{K-1}$$

这种方法有两个问题：首先，为了保持训练集较大，我们允许确认集较小。其次，训练集在相当大程度上重复，确切的说，任意两个训练集有$K-2$份数据重复。

K一般为10或30。当K增加的时候，用于训练的实例的比例增加因而产生更为鲁棒的估计，但是确认集相应变小。此外，也带来了将分类器训练K次的代价，这一代价随着K增加而增加。K-折交叉确认的一个极端情况是N个实例的数据集上的留一(leave-one-out)，其中只保留一个实例作为确认集，其余$N-1$个实例作为训练集。由此我们通过在每次迭代中保留不同的实例而得到N对不同的训练/确认集。这种方法通常用于诸如医疗诊断的应用中，这类应用中很难找到标记数据。留一无法进行分层。

14.2.2 5×2 交叉确认

Dietterich(1998)提出了5×2交叉确认(5×2 cross-validation)，其中使用等大小的训练和确认集。将数据集$\mathcal{X}$随机地分为两部分：$\mathcal{X}_1^{(1)}$和$\mathcal{X}_1^{(2)}$，这样就给出了第一对训练和确认集：$\mathcal{T}_1 = \mathcal{X}_1^{(1)}$和$\mathcal{V}_2 = \mathcal{X}_1^{(2)}$。然后我们交换两个半份的角色来得到第二对训练和确认集：$\mathcal{T}_2 = \mathcal{X}_1^{(2)}$和$\mathcal{V}_2 = \mathcal{X}_1^{(1)}$。这就是第一次对折；$\mathcal{X}_i^{(j)}$表示了第$i$次对折中的第$j$个半份。 331

为了得到第二次对折，我们随机的将 X 打乱并将其划分为新的对折 $X_2^{(1)}$ 和 $X_2^{(2)}$。这可通过从 X 中随机无放回抽样来实现，即，$X_1^{(1)} \cup X_1^{(2)} = X_2^{(1)} \cup X_2^{(2)} = X$。然后对调二者来得到另一对数据集。我们再做第三次对折，因为每次对折我们得到两对数据，做五次对折我们共得到 10 个训练和确认集：

$$\begin{aligned}
\mathcal{T}_1 &= X_1^{(1)} \quad \mathcal{V}_1 = X_1^{(2)} \\
\mathcal{T}_2 &= X_1^{(2)} \quad \mathcal{V}_2 = X_1^{(1)} \\
\mathcal{T}_3 &= X_2^{(1)} \quad \mathcal{V}_3 = X_2^{(2)} \\
\mathcal{T}_4 &= X_2^{(2)} \quad \mathcal{V}_4 = X_2^{(1)} \\
&\vdots \\
\mathcal{T}_9 &= X_5^{(1)} \quad \mathcal{V}_9 = X_5^{(2)} \\
\mathcal{T}_{10} &= X_5^{(2)} \quad \mathcal{V}_{10} = X_5^{(1)}
\end{aligned}$$

当然，我们可以做超过五次对折以获得更多对的训练/确认集，但是 Dietterich(1998)指出在五次对折之后，各集合共享了许多实例，过度的重叠使得由此计算的统计量，确切的说，确认误差率变得相互依赖而无法增加新的信息。即使只是有五次对折，各集合也是有交集而统计量也相互依赖，但是直到五折之前这些影响我们还是可以容忍的。而从另一方面来说，如果使用更少的对折次数，我们获得更少的数据(少于 10 对)，而将无法获得足够大的样本来拟合分布并进行假设检验。

14.2.3 自助法

为了从单个样本中产生多个样本，替代交叉确认的另一个选择是*自助法*(bootstrap)，即从原始样本中以*有放回地*(with replacement)抽取实例的方法来产生新的样本。自助样本可能比交叉确认样本有更多的交集，因而其估计可能更为相互依赖；但对小数据集，这种方法被
332 认为是最好的方法。

在自助法中，我们从大小为 N 的数据集中有放回地抽取 N 个实例。如果只确认一次，则原始数据集作为确认集；否则，我们可以重复多次来产生多个训练/确认集。选取一个实例的概率为 $1/N$；不选取这个实例的概率为 $1-1/N$。N 次抽取均未选取这个实例的概率为：

$$\left(1 - \frac{1}{N}\right)^N \approx e^{-1} = 0.368$$

这意味着训练集包含了大约 63.2% 的实例；也就是说，系统未在 36.8% 的数据上进行训练，因而误差估计是悲观的。解决方法是重复该过程多次并取平均值。

14.3 误差度量

当(损失函数)使用 0/1 损失时，所有的错误都有相同的影响，而我们对误差的计算是基于*混淆矩阵*(confusion matrix)的(表 14-1)。我们可将误差率定义为：

$$\text{误差率} = \frac{|FN| + |FP|}{N} \tag{14.1}$$

表 14-1 混淆矩阵

实际的类	预测的类	
	是	否
是	TP：真正	FN：假负
否	FP：假正	TN：真负

其中 $N = |TP| + |FP| + |TN| + |FN|$ 是确认集中的实例总数。对于采用任意损失函数的一般情况而言，实例总数应该被替换为确认集上的风险(3.3 节)。

为了分析 $K > 2$ 类时的误差，*类混淆矩阵*(class confusion matrix)是很有用的。类混淆矩阵是一个 $K \times K$ 的矩阵，其元素 (i, j) 是属于 C_i 类但却误分到 C_j 类的实例的个数。理想情况下，所有的非对角线元素均应为 0，表示没有误分类。类混淆矩阵允许我们确定误分类发生时的类型，即是否有两个类经常被混淆。

为了调整分类器，另一个方法是绘制*接受者操作特征*(receiver operating characteristic，ROC)曲线，表示命中率和假警报率的比值，即 $|TP|/(|TP| + |FN|)$ 比 $|FP|/(|FP| + |TN|)$，其形式类似于图 14-1。对于每种分类算法，存在一个参数，例如决策阈值，我们可以使用它改变真正与假正比。增加真正的数量也会增加假警报的数量；而降低假警报的次数也会降低命中的次数。基于特定的应用，我们根据这些特征(命中率、假报警率等)的性能/代价情况在这条曲线上确定一个点。

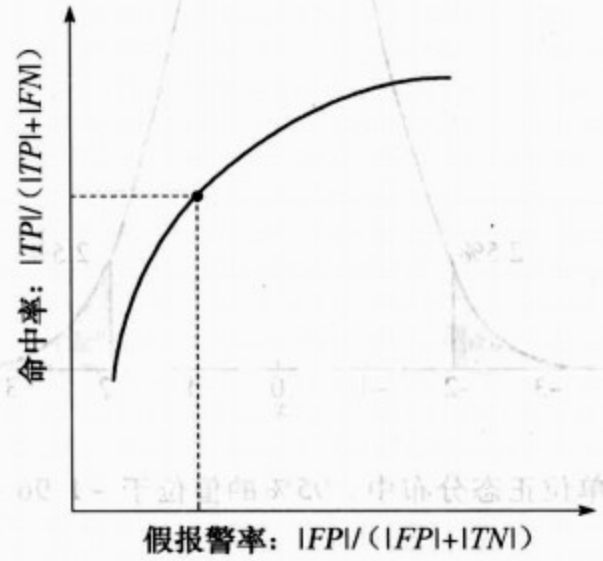


图 14-1 典型的 ROC 曲线。每个分类器有一个参数，例如一个阈值，该参数使得我们可以在曲线上移动，并根据命中和假警报(即真正和假正)之间的相对重要程度来在曲线上确定一个点

14.4 区间估计

下面快速回顾一下我们在假设检验中将用到的*区间估计*(interval estimation)。点估计，如最大似然估计，是对参数 θ 指定一个值。在区间估计中，我们以某种置信度对参数 θ 位于的区间进行确定。为了获取此类区间估计，我们利用点估计的概率分布。

例如，我们要从样本$X=\{x^t\}_{t=1}^N$中估计正态密度的均值μ。$m=\sum_t x^t/N$是样本平均值，并且是对均值的点估计。m是正态分布值之和，因而也是正态的，$m\sim\mathcal{N}(\mu, \sigma^2/N)$。我们以单位正态分布来定义该统计量：

$$\sqrt{N}\frac{(m-\mu)}{\sigma}\sim Z \tag{14.2}$$

我们知道95%的Z落在$(-1.96, 1.96)$中，即，$P\{-1.96<Z<1.96\}=0.95$，因而我们有(见图14-2)

$$P\left\{-1.96<\sqrt{N}\frac{(m-\mu)}{\sigma}<1.96\right\}=0.95$$

或等价地

$$P\left\{m-1.96\frac{\sigma}{\sqrt{N}}<\mu<m+1.96\frac{\sigma}{\sqrt{N}}\right\}=0.95$$

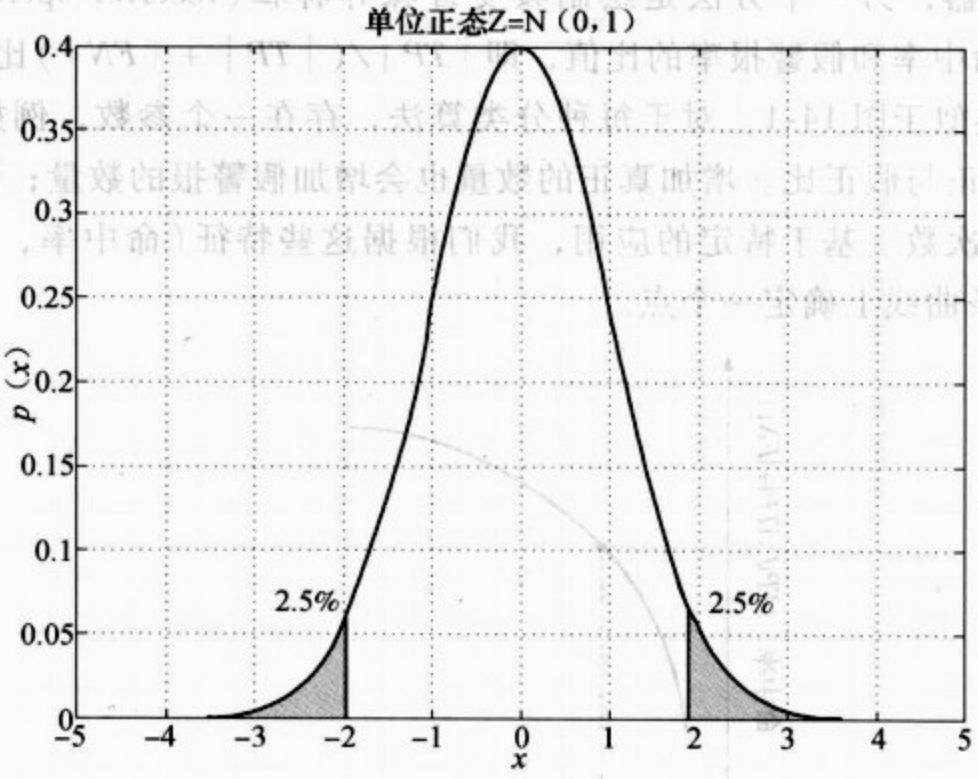


图14-2 在单位正态分布中，95%的值位于-1.96~1.96之间

也就是说“以95%的置信度”，μ落在样本平均值正负$1.96\sigma/\sqrt{N}$个单位的区间内。这即是双侧置信区间(two-sided confidence interval)。以99%的置信度，μ落在$(m-2.85\sigma/\sqrt{N}, m+2.85\sigma/\sqrt{N})$中。也就是说，如果我们需要更高的置信度，则区间增大。随着样本集的规模N变小，区间也增大。

这可以按如下方法推广到任意置信度：令z_α使得

$$P\{Z>z_\alpha\}=\alpha, 0<\alpha<1$$

由于Z是关于均值对称的，所以有$z_{1-\alpha/2}=-z_{\alpha/2}$且$P\{X<-z_{\alpha/2}\}=P\{X>z_{\alpha/2}\}=\alpha/2$。因而，对于任意给定的置信水平$1-\alpha$，我们有

$$P\{-z_{\alpha/2}<Z<z_{\alpha/2}\}=1-\alpha$$

并且

$$P\left\{-z_{\alpha/2}<\sqrt{N}\frac{(m-\mu)}{\sigma}<z_{\alpha/2}\right\}=1-\alpha$$

或

$$P\left\{m-z_{\alpha/2}\frac{\sigma}{\sqrt{N}}<\mu<m+z_{\alpha/2}\frac{\sigma}{\sqrt{N}}\right\}=1-\alpha \tag{14.3}$$

因而，对于任意 α，可以对 μ 计算置信度为 $100(1-\alpha)\%$ 的双侧置信区间。

类似的，如有 $P\{\mathcal{Z}<1.64\}=0.95$，则我们有(见图 14-3)

$$P\left\{\sqrt{N}\frac{(m-\mu)}{\sigma}<1.64\right\}=0.95$$

或

$$P\left\{m-1.64\frac{\sigma}{\sqrt{N}}<\mu\right\}=0.95$$

并且$(m-1.64\sigma/\sqrt{N},\ \infty)$是 μ 的95%的单侧上置信区间(one-sided upper confidence interval)，其定义了一个下界。一般而言，μ 的 $100(1-\alpha)\%$ 的单侧置信区间可通过下式计算

$$P\left\{m-z_{\alpha}\frac{\sigma}{\sqrt{N}}<\mu\right\}=1-\alpha \tag{14.4}$$

类似地，可计算定义上界的单侧下置信区间。

在上述各区间中，我们使用了 σ，即我们假定方差是已知的。如果方差未知，则我们可用样本方差

$$S^2=\sum_t (x^t-m)^2/(N-1)$$

来替代 σ^2。我们知道当 $x^t\sim\mathcal{N}(\mu,\ \sigma^2)$时，$(N-1)S^2/\sigma^2$ 是自由度为 $N-1$ 的卡方(分布)。我们同样知道 m 和 S^2 是相互独立的。于是，$\sqrt{N}(m-\mu)/S$ 是自由度为 $N-1$ 的 t 分布(A.3.7 节)，记作

$$\frac{\sqrt{N}(m-\mu)}{S}\sim t_{N-1} \tag{14.5}$$

因而，对任意的 $\alpha\in(0,\ 1/2)$，我们可以使用该 t 分布(t distribution)，而非单位正态分布$\mathcal{Z}$确定的值来定义一个区间

$$P\left\{t_{1-\alpha/2,N-1}<\sqrt{N}\frac{(m-\mu)}{S}<t_{\alpha/2,N-1}\right\}=1-\alpha$$

或使用 $t_{1-\alpha/2,N-1}=-t_{\alpha/2,N-1}$而有

$$P\left\{m-t_{\alpha/2,N-1}\frac{S}{\sqrt{N}}<\mu<m+t_{\alpha/2,N-1}\frac{S}{\sqrt{N}}\right\}=1-\alpha$$

类似地，可定义单侧置信区间。t 分布比单位正态分布有更大的展宽(较长的尾)，因而 t 分布给出的区间一般更大；考虑到未知方差所引入的附加的不确定性的存在，这应该在预料之中。

335
~
336

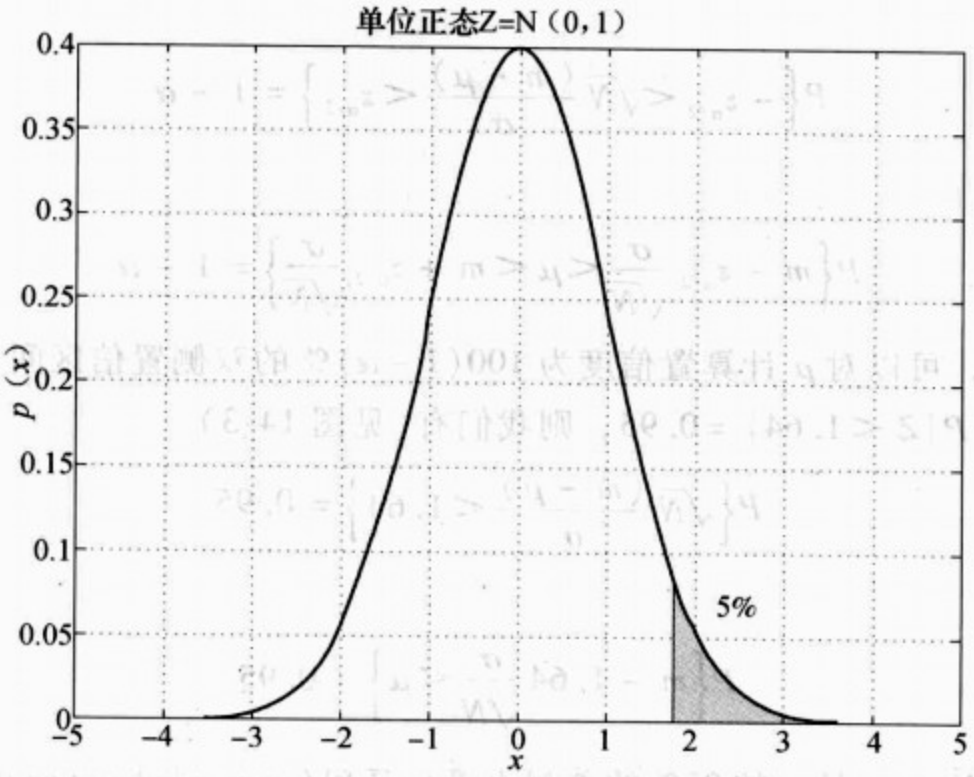


图 14-3 单位正态分布的 95% 落在 1.64 之前

14.5 假设检验

在某些应用中，我们可能希望使用样本对涉及参数的一些特定假设进行检验，而不是显式地估计某些参数。例如，我们可能希望检验均值是否小于 0.02，而不是估计均值。如果随机样本与所考虑的假设一致，则我们说该假设被“接受”，否则，我们说它被“拒绝”。但是，当我们做这样的决定时，我们并非说假设为真，而是说在一定的置信程度上，样本数据和假设看起来是一致的。

在假设检验(hypothesis testing)中，方法如下：我们定义一个服从某一分布的统计量来判断假设是否正确。如果从样本中计算的统计量具有足够高的概率是取自该分布，则接受该假设；否则拒绝它。

假设有一个样本取自一个均值 μ 未知、方差 σ^2 已知的正态分布，而我们希望对一个关于 μ 的假设进行检验，如，其值是否等于一个指定的值 μ_0。该假设记作 H_0 并称之为原假设(null hypothesis)

$$H_0 : \mu = \mu_0$$

相对的备择假设为

$$H_1 : \mu \neq \mu_0$$

m 是 μ 的点估计，而且当 m 和 μ_0 相去不远的时候，接受 H_0 是合理的。这正是要使用区间估计的地方：我们以显著水平(level of significance) α 接受该假设，如果 μ_0 位于 $100(1-\alpha)\%$ 的置信区间，即 H_0 被接受，如果

$$\frac{\sqrt{N}(m-\mu_0)}{\sigma} \in (-z_{\alpha/2}, z_{\alpha/2}) \tag{14.6}$$

这是一个双侧检验(two-sided test)。如果当假设是正确的时候我们拒绝了它，这就是一个第一类错误(type Ⅰ error)，而在检验之前设定的 α 值定义了我们可以在多大程度上容忍

第一类错误，通常取值为：$\alpha=0.1$，0.05，0.01（见表 14-2）。第二类错误（type Ⅱ error）是如果在真实均值 μ 和 μ_0 不相等的情况下我们接受了原假设。当真实均值为 μ 时接受 H_0 的概率是 μ 的函数，由下式给出

$$\beta(\mu) = P_{\mu}\left\{-z_{\alpha/2} \leqslant \frac{m-\mu_0}{\sigma/\sqrt{N}} \leqslant z_{\alpha/2}\right\} \tag{14.7}$$

337
≀
338

$1-\beta(\mu)$ 称为检验的*功效函数*（power function），并且等于当 μ 为真实均值时假设被拒绝的概率。

表 14-2　第一类错误、第二类错误和检验功效

	决　策	
事　实	接　受	拒　绝
真	正　确	第一类错误
假	第二类错误	正确（功效）

与备择假设为 $\mu \neq \mu_0$ 时的双侧检验相反，也可以进行如下形式的单侧检验（one-sided test）：

$$H_0: \mu \leqslant \mu_0$$
$$H_1: \mu > \mu_0$$

显著性水平为 α 的单侧检验定义了界定于单侧的 $100(1-\alpha)\%$ 置信区间，为了接受假设，m 的值必须位于该区间内。我们接受该假设，如果

$$\frac{\sqrt{N}}{\sigma}(m-\mu_0) \in (-\infty, z_{\alpha}) \tag{14.8}$$

如果方差未知，我们可以像在区间估计中所做的那样，以样本方差来替代总体方差并利用下述事实

$$\frac{\sqrt{N}(m-\mu_0)}{S} \sim t_{N-1} \tag{14.9}$$

例如，对于 $H_0: \mu=\mu_0$ 和 $H_1: \mu \neq \mu_0$，我们以显著性水平 α 接受假设，如果

$$\frac{\sqrt{N}(m-\mu_0)}{S} \in (-t_{\alpha/2,N-1}, t_{\alpha/2,N-1}) \tag{14.10}$$

这就是*双侧 t 检验*（two-sided t test）。相似的可以定义单侧 t 检验。

14.6　评估分类算法的性能

现在，我们已经回顾了假设检验，为理解如何将其应用于误差率检验做好了准备。我们从误差率评估入手，在下一节讨论误差率比较。 339

14.6.1　二项检验

我们从只有一个训练集 $\mathcal{T}$ 和一个确认集 $\mathcal{V}$ 的情形入手。我们在 $\mathcal{T}$ 上训练分类器并在 $\mathcal{V}$ 上检验。我们以 p 表示分类器误分类的概率。p 是未知的；因而我们要对它进行估计或对关于

它的假设进行检验。对于来自确认集$\mathcal{V}$的索引号为t的实例，我们以x^t表示分类决策的正确性。这样的话，x^t取值为0/1；x^t是伯努利分布，分类器以概率p产生一次误分类并且x^t取1值，以概率$1-p$正确分类并且x^t取0值。对p值的点估计为(4.2.1节)

$$\hat{p} = \frac{\sum_t x^t}{N} \tag{14.11}$$

其中$N = |\mathcal{V}|$。现在我们想要检验错误概率p是否小于或等于我们设定的某个值p_0。问题可以表述为：假定一个分类器在一个大小为N的确认集上有e次误分类，是否可以说分类器的误差概率为p_0或更小？

我们有如下假设检验

$$H_0 : p \leqslant p_0 \text{ 对 } H_1 : p > p_0$$

令X表示在大小为N的确认集上的误分类次数：

$$X = \sum_{t=1}^{N} x^t$$

因为x^t是独立的、服从伯努利分布的随机变量，因此其和X服从二项分布。如果错误概率为p，则分类器在N次分类中有j次误分的概率为

$$P\{X = j\} = C_N^j p^j (1-p)^{N-j}$$

在原假设下，我们假定p(最多)为p_0，而最多有e次误分类的概率为

$$P\{X \leqslant e\} = \sum_{j=0}^{e} C_N^j p_0^j (1-p_0)^{N-j} \tag{14.12}$$

如果这个概率小于所允许的概率$1-\alpha$，二项检验(binomial test)接受该假设；否则，拒绝它。

340

14.6.2 近似正态检验

二项检验计算代价较高。因为X是服从相同分布的独立随机变量之和，依据中心极限定理，对于大的N值和小的p_0，X近似服从均值为Np_0、方差为$Np_0(1-p_0)$的正态分布。于是

$$\frac{X - Np_0}{\sqrt{Np_0(1-p_0)}} \sim \mathcal{Z} \tag{14.13}$$

其中~表示“近似服从分布”。于是，当$X=e$时，如果上式的值小于或等于$z_{1-\alpha}$，则近似正态检验(approximate normal test)接受原假设。其中$z_{0.95}$的值是1.64。当$Np_0 > 20$时，该检验可能给出错误的结果。

14.6.3 配对t检验

前述两种检验方法都使用一个确认集。如果在K对训练/确认集上运行算法K次，则我们在K个确认集上得到K个错误百分比p_i，$i=1, \cdots, K$。设x_i^t为1如果在$\mathcal{T}_i$上训练的分类器对$\mathcal{V}_i$中的实例t产生了一次误分；否则x_i^t为0。于是

$$p_i = \frac{\sum_{t=1}^{N} x_i^t}{N}$$

由于有

$$m = \frac{\sum_{i=1}^{K} p_i}{K}, \qquad S^2 = \frac{\sum_{i=1}^{K} (p_i - m)^2}{K-1}$$

根据(14.9)式，我们有

$$\frac{\sqrt{K}(m - p_0)}{S} \sim t_{K-1} \tag{14.14}$$

如果上式的值小于或等于 $t_{\alpha,K-1}$，则配对 t 检验接受原假设：分类算法以显著性水平 α 具有 p_0 或更低的错误率。通常，K 取值为 10 或 30。$t_{0.05,9} = 1.83$ 而 $t_{0.05,29} = 1.70$。

14.7 比较两个分类算法

给定两个学习算法和一个训练集，我们想要比较和检验这两个算法所构建的分类器在新的实例上是否具有相同的期望误差率。

14.7.1 McNemar 检验

给定一个训练集和一个确认集，我们使用两个算法在训练集上训练两个分类器并在确认集上对它们进行检验并计算它们的误差。如下所示的列联表(contingency table)是一个矩阵形式的自然数数组，表示各种情况的计数或频率：

e_{00}：两个分类器均误分的实例个数	e_{01}：分类器 1 误分而分类器 0 没有误分的实例个数
e_{10}：分类器 2 误分而分类器 1 没有误分的实例个数	e_{11}：两个分类器均正确分类的实例个数

在两个分类算法有相同的错误率的原假设下，我们期望 $e_{01} = e_{10}$，二者的值均为 $(e_{01} + e_{10})/2$。我们有自由度为 1 的卡方统计量

$$\frac{(|e_{01} - e_{10}| - 1)^2}{e_{01} + e_{10}} \sim \chi_1^2 \tag{14.15}$$

并且如果上述统计量小于或等于 $\chi_{\alpha,1}^2$，则 McNemar 检验接受假设：两个分类算法以显著水平 α 具有相同的误差率。$\chi_{0.05,1}^2 = 3.84$。

14.7.2 K-折交叉确认配对 t 检验

使用 K-折交叉确认在数据集上产生 K 对训练/确认集。我们使用两个分类算法在训练集 $\mathcal{T}_i (i = 1, \cdots, K)$ 上训练并在各确认集 $\mathcal{V}_i$ 上检验。两个分类器在确认集上的误差率分别记作 $p_i^{(1)}$ 和 $p_i^{(2)}$。如果两个分类算法具有相同的误差率，则我们期望它们具有相同的均值，或等

价地说，它们的均值之差为0。

在第 i 折，两个分类器的误差率之差是 $p_i = p_i^{(1)} - p_i^{(2)}$。$K$ 次比较之后，得到一个包含 K 个点的 p_i 的分布。由于 $p_i^{(1)}$ 和 $p_i^{(2)}$ 二者是(近似)正态的，其差 p_i 也是正态的。原假设为该分布的均值为0：

$$H_0 : \mu = 0$$
$$H_1 : \mu \neq 0$$

我们定义

$$m = \frac{\sum_{i=1}^{K} p_i}{K}, \qquad S^2 = \frac{\sum_{i=1}^{K} (p_i - m)^2}{K-1}$$

341
~
342

在 $\mu = 0$ 的原假设下，我们有一个自由度为 $K-1$ 的统计量：

$$\frac{\sqrt{K}(m-0)}{S} = \frac{\sqrt{K} \cdot m}{S} \sim t_{K-1} \tag{14.16}$$

因而，如果该值落在区间 $(-t_{\alpha/2,K-1}, t_{\alpha/2,K-1})$，则 K-折交叉确认配对 t 检验(K-fold cv paired t test)接受假设：两个分类算法以显著水平 α 具有相同的误差率。$t_{0.025,9} = 2.26$，$t_{0.025,29} = 2.05$。

14.7.3 5×2 交叉确认配对 t 检验

在 Dietterich(1998)提出的 5×2 交叉确认 t 检验中，我们进行五轮对折交叉确认。在每轮中，数据集被划分为等大小的集合。$p_i^{(j)}$ 表示两个分类器在第 i 轮中第 j 折数据上的误差率之差，其中 $i = 1, \cdots, 5$，$j = 1, 2$。在第 i 轮的平均值为 $\bar{p}_i = (p_i^{(1)} + p_i^{(2)})/2$，估计方差为 $s_i^2 = (p_i^{(1)} - \bar{p}_i)^2 + (p_i^{(2)} - \bar{p}_i)^2$。

在两个分类算法具有相同的误差率的原假设下，$p_i^{(j)}$ 是两个同分布的比例值之差，而忽略这些比例值不是相互独立的事实，$p_i^{(j)}$ 可近似的被认为服从均值为0，方差 σ^2 未知的正态分布。于是，$p_i^{(j)}/\sigma$ 是近似单位正态的。如果假定 $p_i^{(1)}$ 和 $p_i^{(2)}$ 是独立、正态的(严格地说并非如此，因为它们的训练和确认集并非相互独立抽取的)，则 s_i^2/σ^2 服从自由度为1的卡方分布。如果假定每个 s_i^2 是相互独立的(事实并非如此因为它们从相同的数据集计算得到)，则它们的和服从自由度为5的卡方分布：

$$M = \frac{\sum_{i=1}^{5} s_i^2}{\sigma^2} \sim \chi_5^2$$

并且

$$t = \frac{p_1^{(1)}}{\sqrt{M/5}} = \frac{p_1^{(1)}}{\sqrt{\sum_{i=1}^{5} s_i^2/5\sigma^2}} \sim t_5 \tag{14.17}$$

上式给出了自由度为5的 t 统计量。如果该统计量的值落在区间 $(-t_{\alpha/2,5}, t_{\alpha/2,5})$，则 5×2 交叉确认 t 检验(5×2 cv paired t test)接受假设：两个分类算法以显著性水平 α 具有相同的误差率。$t_{0.025,5} = 2.57$。

我们也可以定义一个单侧检验来检查第一个分类算法的期望误差率是否小于或等于第二个算法，即：

$$H_0 : \mu \leqslant 0$$
$$H_1 : \mu > 0$$

我们计算和(14.17)式相同的统计量并在它小于 $t_{\alpha,5}$ 时接受原假设。$t_{0.05,5} = 2.02$。

14.7.4 5×2 交叉确认配对 F 检验

我们注意到(14.17)式中的分子 $p_i^{(1)}$ 是任意的；实际上，有 10 个不同的值可当作分子，即：$p_i^{(j)}$，$j=1$，2，$i=1$，…，5，产生 10 个可能的统计量：

$$t_i^{(j)} = \frac{p_i^{(j)}}{\sqrt{\sum_{i=1}^{5} s_i^2/5}} \tag{14.18}$$

Alpaydın(1999)提出了 5×2 交叉确认 t 检验的扩展，组合 10 个可能的统计量的结果。如果 $p_i^{(j)}/\sigma \sim \mathcal{Z}$，则有 $(p_i^{(j)})^2/\sigma^2 \sim \chi_1^2$，并且其和是自由度为 10 的卡方分布：

$$N = \frac{\sum_{i=1}^{5}\sum_{j=1}^{2}(p_i^{(j)})^2}{\sigma^2} \sim \chi_{10}^2$$

将上式作为(14.17)式的分子，得到的统计量为两个卡方分布随机变量的比值。两个变量分别除以它们的自由度得到第一自由度为 10，第二自由度为 5 的 F 分布(A.3.8 节)：

$$f = \frac{N/10}{M/5} = \frac{\sum_{i=1}^{5}\sum_{j=1}^{2}(p_i^{(j)})^2}{2\sum_{i=1}^{5} s_i^2} \sim F_{10,5} \tag{14.19}$$

如果该值小于 $F_{\alpha,10,5}$，则 5×2 交叉确认配对 F 检验(5×2cv paired F test)接受假设：两个分类算法以显著性水平 α 具有相同的误差率。$F_{0.05,10,5} = 4.74$。

343
≀
344

14.8 比较多个分类算法：方差分析

在很多情况下，我们有多个候选分类算法，我们希望找到最准确的一个。给定 L 个候选分类算法，我们在 K 个训练集上对其进行训练，每个算法产生 K 个分类器，而后在 K 个确认集上进行检验并记录相应的误差率。这样产生了 L 组，每组 K 个误差值。于是，问题是比较这 L 个样本的差异的统计显著性。

在*方差分析*(analysis of variance)中，我们考虑 L 个独立的样本，每个大小为 K，由未知均值 μ_j 和未知公共方差 σ^2 的正态随机变量组成：

$$X_{ij} \sim \mathcal{N}(\mu_j, \sigma^2), j = 1, \cdots, L, i = 1, \cdots, K,$$

并且我们想对“所有均值相等”的假设 H_0 进行检验：

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_L$$

对多个分类算法的误差率进行比较就属于这种情况。对 L 个分类算法，我们有其在 K 个确认折上的误差率。X_{ij} 是分类算法 j 在第 i 折数据所训练的分类器的确认错误次数。每个

X_{ij}是二项式的并近似服从正态分布。如果接受 H_0，则我们可以断言 L 个分类算法之间的误差率不存在显著差异。因而，这是我们在 14.7 节所看到的对两个分类算法进行比较的检验方法的推广。这 L 个分类算法可能不同或使用不同的超参数，如多层感知器的隐藏单元数，k-nn的近邻数等等。

方差分析方法是导出两个关于 σ^2 的估计。第一个估计只有在 H_0 为真的时候才为真，而第二个估计始终是一个有效估计，无论 H_0 是否为真。如果两个估计显著不同，则方差分析拒绝 H_0，即 L 个样本取自相同的总体。

第一个关于 σ^2 的估计是有效的，仅当假设为真，即 $\mu_j = \mu$，$j = 1, \cdots, L$。如果 $X_{ij} \sim \mathcal{N}(\mu, \sigma^2)$，则各组平均

345

$$m_j = \sum_{i=1}^{K} \frac{X_{ij}}{K}$$

也是正态的，均值为 μ、方差为 σ^2/K。如果假设为真，则 $m_j (j = 1, \cdots, L)$ 是 L 个取自 $\mathcal{N}(\mu, \sigma^2/K)$ 的实例。于是，它们的均值和方差分别为

$$m = \frac{\sum_{j=1}^{L} m_j}{L}, S^2 = \frac{\sum_j (m_j - m)^2}{L - 1}$$

因而，σ^2 的一个估计是 $K \cdot S^2$，即

$$\hat{\sigma}^2 = K \sum_{j=1}^{L} \frac{(m_j - m)^2}{L - 1} \tag{14.20}$$

每个 m_j 都是正态，并且 $(L-1)S^2/(\sigma^2/K)$ 是自由度为 $(L-1)$ 的卡方分布。因而，我们有

$$\sum_j \frac{(m_j - m)^2}{\sigma^2/K} \sim \chi^2_{L-1} \tag{14.21}$$

我们定义组间平方和 SS_b 为

$$SS_b \equiv K \sum_j (m_j - m)^2$$

因而，当 H_0 为真时，我们有

$$\frac{SS_b}{\sigma^2} \sim \chi^2_{L-1} \tag{14.22}$$

第二个关于 σ^2 的估计是组方差 S_j^2 的平均值。组方差定义为

$$S_j^2 = \frac{\sum_{i=1}^{K} (X_{ij} - m_j)^2}{K - 1}$$

而它们的平均值为

$$\hat{\sigma}^2 = \sum_{j=1}^{L} \frac{S_j^2}{L} = \sum_j \sum_i \frac{(X_{ij} - m_j)^2}{L(K-1)} \tag{14.23}$$

我们定义组内平方和 SS_w 为

$$SS_w \equiv \sum_j \sum_i (X_{ij} - m_j)^2$$

回忆一下，对正态样本，我们有

$$(K-1)\frac{S_j^2}{\sigma^2} \sim \chi^2_{K-1}$$

346

并且卡方分布之和仍然是卡方分布，我们有

$$(K-1)\sum_{j=1}^{L}\frac{S_j^2}{\sigma^2} \sim \chi^2_{L(K-1)}$$

因而

$$\frac{SS_w}{\sigma^2} \sim \chi^2_{L(K-1)} \tag{14.24}$$

如果两个估计显著不同，则我们应当拒绝 H_0。而如果 H_0 不为真，则第一个估计高估了 σ^2。两个独立卡方变量分别除以其相应的自由度的比值是一个服从 F 分布的随机变量，因而当 H_0 为真时，我们有

$$\left(\frac{SS_b/\sigma^2}{L-1}\right)\Big/\left(\frac{SS_w/\sigma^2}{L(K-1)}\right) = \frac{SS_b/(L-1)}{SS_w/(L(K-1))} \sim F_{L-1,L(K-1)} \tag{14.25}$$

对于任意给定的显著性水平值 α，如果该统计量小于 $F_{\alpha,L-1,L(K-1)}$，则接受 L 个分类算法具有相同的期望误差率的假设。这就是基本的单向方差分析，其中只有单个因素，例如，分类算法。

如果假设被拒绝，我们只是知道 L 个分组之间存在某种差异；我们并不知道分类算法的误差率是怎样的不同。为此，我们可以在分类算法的子集上进行方差分析来确定误差率相当的子集。我们不需要考虑所有可能的子集；我们可以将分类算法按照平均误差排序，然后对相继的分类算法进行检验。为了找到最大的分组，我们从大子集向小子集进行：首先对所有 L 个分组进行检验；如果被拒绝，则对两个 $L-1$ 的子集(去掉第一个或最后一个)进行检验，等等。最后，我们得到没有显著差异的分组。例如，我们的结果可以是145 23，表示有两个分组，一个由分类算法 1、4、5 组成，另一个由 2 和 3 组成。

也存在非参数检验来进行对照(contrast)检查(Dean and Voss 1999)：假设算法 1 和 2 是参数方法，而算法 3 和 4 是非参数方法。我们可以检验算法 1 和 2 的平均是否不同于 3 和 4 的平均。

我们还可以使用 14.7 节中讨论的各种检验方法进行一系列的两两比较来检验两两差异。在统计学中，这称为多重比较(multiple comparison)。然而，如果我们在使用一组检验之后 347
做出决定，有一点需要注意：如果要以显著性水平 α 对 m 个假设进行检验，则至少有一个假设不正确地被拒绝的概率至多为 $m\alpha$。例如，每个均以 95% 的单个置信区间进行计算的 6 个置信区间同时正确的概率至少为 70%。因而，为了确保整体置信区间至少为 $100(1-\alpha)$，单个置信度区间应当为 $100(1-\alpha/m)$。这称之为 Bonferroni 校正(Bonferroni correction)。

在 K 个训练/确认集上对 L 个分类算法进行训练和检验是主要的代价所在。一旦训练、检验完成并且结果存储在一个 $K \times L$ 的表中，从中进行方差分析或计算两两比较检验统计量相对而言就廉价许多。

14.9 注释

关于区间估计，假设检验和方差分析的更为详细的讨论可以在任何统计学导论书籍中找到，如 Ross1987。Dietterich(1998)讨论了各种统计检验方法，并在多个应用上使用不同的分

类算法对其进行了比较。Jensen 和 Cohen(2000)讨论了如何优化学习器的超参数。

在比较两个分类算法方面，请注意我们只是对它们是否具有相同的期望误差率进行检验。如果是，这也不意味它们产生相同的错误。这是我们在第 15 章将要讨论的想法；如果不同的分类器产生不同的错误，则我们可以通过组合多个模型来提高准确率。

另一个需要注意的要点是，我们只对误分类率进行评估或比较。这意味着从我们的观点出发，所有的误分类都具有相同的代价。如果事实并非如此，则我们的检验应当基于风险，将一个合适的损失函数考虑在内。这一方面的工作还不是很多。类似的，这些检验也应当从分类推广到回归，使得可以对回归算法的均方误差进行评估(4.6 节)，或可以对两个回归算法的误差进行比较。

我们所讨论的是参数检验，因为我们假定了某种参数模型并在其上定义了假设，例如
348 $H_0: \mu = 0$。也有非参数检验(nonparametric test)(Conovar1999)。例如，Kruskal-Wallis 检验即是方差分析的非参数版本，其中给定了一定数量的样本，每个来自一个总体，而我们要对所有的数据源是相同的原假设进行检验。Newman-Keuls 检验是一种非参数范围检验，用来寻找误差率相当的子集并对其排序；比如，用它可以发现诸如145 23 这样的排序。多重比较的方法在 Dean 和 Voss1999 中讨论。

Statlog 项目(Michie、Spiegelhalter 和 Taylor1994)对二十个不同的分类算法在大数量应用上进行了比较。另一个是 *Delve* 项目，允许研究者增加新的数据集和分类算法来和其他的算法比较(Hinton 和 Delve 小组成员 1995)。

当我们比较两个或更多算法的时候，如果它们具有相同误差率的原假设被接受，我们选用最简单，即空间和时间复杂度最小的算法。也就是说，如果数据在误差率方面并不偏好任何一个分类算法的时候，我们使用先验知识的偏好。例如，如果我们对一个线性模型和一个非线性模型进行比较，并且检验接受了二者具有相同的误差率，则我们将选择更为简单的线性模型。即使检验拒绝了这样的假设，在选择算法的时候，错误率也仅仅是一个标准。其他如训练(空间/时间)复杂度、检验复杂度和可解释性在实际应用中都可能是更重要的标准。

14.10 习题

1. 我们可以通过从一个伯努利分布抽取样本来模拟一个误差概率为 p 的分类器。进行此模拟，并对 $p_0 \in (0, 1)$进行二项检验，近似正态检验和 t 检验。对不同的 p 值，将这些检验进行至少 1000 次并计算拒绝原假设的概率。当 $p_0 = p$ 时，你认为拒绝的期望概率如何？
2. 假设 $x^t \sim \mathcal{N}(\mu, \sigma^2)$，其中 σ^2 已知。对假设 $H_0: \mu \geq \mu_0$ 和 $H_1: \mu < \mu_0$ 如何进行检验？
3. K-折交叉确认 t 检验只对误差率相等的假设进行检验。如果假设被拒绝，我们并不知道哪个分类算法具有更低的误差率。我们如何对第一个分类算法不比第二个分类算法的误差率更高的假设进行检验？提示：需要对 $H_0: \mu \leq 0$ 和 $H_1: \mu > 0$ 进行检验。
349 4. 假设有三个分类算法。如何将其从最好到最差进行排序？

14.11 参考文献

Alpaydın, E. 1999. "Combined 5 × 2cv*F* Test for Comparing Supervised Classification Learning Algorithms." *Neural Computation* 11: 1885 – 1892.

Conover, W. J. 1999. *Practical Nonparametric Statistics*. 3rd ed. New York: Wiley.

Dean, A., and D. Voss. 1999. *Design and Analysis of Experiments*. New York: Springer.

Dietterich, T. G. 1998. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms." *Neural Computation* 10: 1895 – 1923.

Hinton, G. E., and Delve Team Members. 1995. *Assessing Learning Procedures Using Delve*. http://www.cs.utoronto.ca/neuron/delve/delve.html.

Jensen, D. D., and P. R. Cohen. 2000. "Multiple Comparisons in Induction Algorithms." *Machine Learning* 38: 309 – 338.

Michie, D., D. J. Spiegelhalter, and C. C. Taylor. 1994. *Machine Learning, Neural and Statistical Classification*. New York: Ellis Horwood.

Ross, S. M. 1987. *Introduction to Probability and Statistics for Engineers and Scientists*. New York: Wiley.

Turney, P. 2000. "Types of Cost in Inductive Concept Learning." Paper presented at *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, Stanford University, Stanford, CA, July 2.

Wolpert, D. H. 1995. "The Relationship between PAC, the Statistical Physics Framework, the Bayesian Framework, and the VC Framework." *In The Mathematics of Generalization*, ed. D. H. Wolpert, 117 – 214. Reading, MA: Addison-Wesley.

第15章　组合多学习器

我们在前面的章节中讨论了许多不同的学习算法。尽管一般而言它们是成功的，但没有哪一个算法总是最准确的。现在，我们将讨论由多个学习器组成的模型。这些学习器互补，因此通过组合它们，我们可以获得更高的准确率。

15.1　基本原理

在任何应用中，我们可以使用多个学习算法中的一个，而使用某算法，存在对最终学习器产生影响的超参数。例如，在分类情况下，我们可以使用参数分类器或多层感知器，而比如对一个多层感知器，我们还要确定隐藏单元的数目。“没有免费的午餐”法则表明没有一个学习算法可以在任何领域总是产生最准确的学习器。通常的方法是试验很多种算法，然后选择在一个在单独的确认集上性能最佳的算法，正如在第14章所讨论的那样。

每一个学习算法都构建了一个基于一组假设的某种模型。当假设在数据上不成立时，这种归纳偏倚将导致误差。学习是一个不适定问题，并且在有限的数据上，每个学习算法都收敛到不同的解，并在不同的情况下失效。可以通过性能调节使一个学习算法在确认集上达到尽可能最高的准确率，但是调节本身就是一个复杂的任务，并且即使对最好的学习器而言也存在实例使其不能足够准确。解决之道在于也许存在另一种学习方法，在这些实例上是准确的。通过合适方式将多个学习器组合可以提高准确率。近来，随着计算和存储变得更为廉价，组合多个学习器的系统也随之流行。

组合总是采取类似决策的学习器是没有意义的，因此我们的目标是可以寻找一组基学习器(base-learner)，它们采取不同的决策以至可以相互补充。为了达到这一目标，我们可在不同的“环节”操作：

1. 最简单的方法是使用不同的学习算法(different learning algorithm)来训练，得到不同的基学习器。不同的算法对数据做不同的假设并产生不同的分类器。比如，一个基学习器可能是参数化的而另一个是非参数的。当我们决定使用一个算法的时候，我们是将重点放在单一方法上并忽略所有其他方法。通过基于多个算法来组合多个学习器，我们将自己从只能接受一个单一决策的境况中摆脱出来，并再也不将所有鸡蛋放在同一个篮子中。

2. 我们可以使用相同的学习算法，但使用不同的超参数(different hyperparameter)。这样的例子包括：多层感知器中的隐藏单元数目，k-最近邻中的k值，决策树中的误差阈值等等。对高斯参数分类器而言，协方差矩阵是否共享是一个超参数。如果在优化算法中使用诸如梯度下降这样的最终状态依赖于初始状态的迭代过程，如使用向后传播的多层感知器，初始状态(如初始权重)是另一种超参数。当我们用不同的超参数值训练多个基学习器时，我们对其取平均值来降低方差，从而减小误差。

3. 不同的基学习器也可以使用相同输入对象或事件的不同表示(different representation)，

从而使得集成不同类型的感知器/测量或特征成为可能。不同的表示凸显了对象的不同特征，从而产生更好的识别。在许多应用中，存在多个信息源，如能使用所有这些数据来提取更多的信息、并在预测中到达更高的准确率是令人期望的。例如，在语音识别中，为了识别语音单词，除了声学输入之外，我们还可以使用讲话者在说这个单词时嘴唇的视频图像。这类似于传感器融合(sensor fusion)，其中来自不同传感器的数据集成在一起，为特定应用提取更多的信息。最简单的方法就是连接所有数据向量并将其当作是来自同一数据源的一个大向 352
量，但是这种方法在理论上似乎不太合适，因为这样相当于对取自多元统计分布的数据进行建模。此外，更高的输入维度使得系统更为复杂，并且需要更大的样本以使估计更准确。我们采用的方法是使用不同的基学习器在不同的数据源上分别进行预测，然后组合这些预测结果。

4. 另一可能的方法是使用不同的训练集(different training set)训练不同的基学习器。这可以通过在给定的样本上随机地抽取训练集来实现；这称之为“装袋”[⊖]。抑或，可以串行地训练学习器，使得前一个基学习器上预测不准的实例在之后的基学习器的训练中获得更多的重视；这种例子有提升(boosting)和级联(cascading)，这些方法积极地尝试生成互补的学习器，而不是靠碰运气。训练样本的划分也可以基于数据空间的局部性来完成，以使每个基学习器在属于输入空间中某一局部的实例上训练；这即是由我们在第 12 章讨论的“混合专家模型”所做的，不过我们从多学习器组合的角度来讨论。类似的，可以将主任务定义为由基学习器实现的若干子任务，如纠错输出码(error-correcting output code)所做的那样。

非常重要的一点是当生成多个基学习器时，只要它们有合理的准确率即可，而不要求它们每个都非常的准确，因此不要，也不需要对这些基学习器进行单独优化以获取最佳准确率。基学习器的选择并不是由于其准确性，而是由于其简单性。然而，我们的确要求基学习器在不同实例上是准确的，专注于问题的子领域。我们所关心的是基学习器在组合后的准确性，而非开始时各基学习器的准确性。例如，我们有一个准确率为 80% 的分类器。当我确定第二个分类器的时候，我们不关心其总体准确率；只要我们知道何时使用哪个分类器，我们只关心其在第一个分类器误分的 20% 实例上的准确性如何。

除了如何训练学习器之外，还存在不同的多个基学习器组合方法来产生最终的输出：

- 多专家组合(multiexpert combination)方法让基学习器并行(parallel)工作。所有基学习器都被训练，然后给定一个实例，所有的基学习器都给出它们的决策，而一个另外的组合器使用它们的预测计算最终的决策。这种方法的例子包括投票(voting)及其变种，混合专家模型(mixture of expert)和层叠泛化(stacked generalization)。 353
- 多级组合(multistage combination)方法使用一种顺序(serial)方法，其中下一个基学习器只在前一个基学习器预测不够准确的实例上进行训练或检验。其基本思想是基学习器(或其所使用的不同表示)按复杂度递增排序，使得除非前一个更简单的基学习器(的结果)不足够可信，否则就不使用复杂的基学习器(或不提取其复杂表示)。一个这样的例子是级联(cascading)。

⊖ bagging 是 bootstrap aggregation(自助聚集)的意思。许多文献直译为“装袋”。我们沿用这种译法。——译者注

假设有 L 个基学习器。我们用 $d_j(x)$ 表示基学习器 $\mathcal{M}_j$ 在给定的任意维输入 x 上的预测结果。在存在多种输入数据表示的情况下，每个 $\mathcal{M}_j$ 使用一个不同的输入表示 x_j。最后的预测从各个基学习器的预测计算：

$$y = f(d_1, d_2, \cdots, d_L \mid \Phi) \tag{15.1}$$

其中 $f(\cdot)$ 是一个组合函数，Φ 表示其参数。当有 K 个输出的时候，每个学习器有 K 个输出 $d_{ji}(x)$，$i=1, \cdots, K$，$j=1, \cdots, L$，而组合它们，我们仍然产生 K 个值 y_i，$i=1, \cdots, K$。进而，比方说在分类中，我们选择具有最大 y_i 值的类(作为分类结果)。

15.2 投票法

组合多个学习器的最简单方法是通过投票(voting)，这相当取学习器的线性组合。这种方法也称集成(ensemble)或线性判断组合(linear opinion pool)。我们以 w_j 表示学习器 j 的权重。那么最终的输出可用下式计算(见图 15-1)

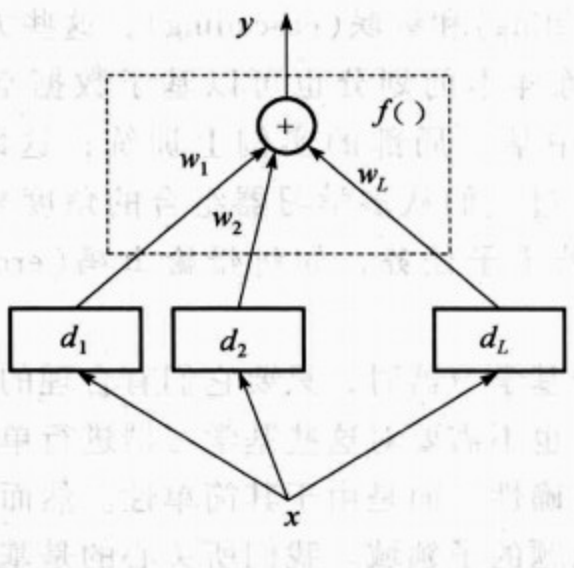


图 15-1 在投票法中，组合函数 $f(\cdot)$ 是一个加权和。d_j 是各个学习器，而 w_j 是它们投票的权重。y 是总体输出。在有多个输出的情况下，比如分类，学习器具有多个输出 d_{ji}，其加权和为 y_i。还要注意，图中所有学习器观测相同的输入；可能不同的学习器观测相同输入对象或事件的不同表示

$$y = \sum_{j=1}^{L} w_j d_j \tag{15.2}$$

满足

$$\text{对任意的 } j, w_j \geqslant 0, \text{并且} \sum_{j=1}^{L} w_j = 1$$

这里，(15.1)式中的 $f(\cdot)$ 相当于一个加权求和函数，其中 Φ 是权重集 $w_1, \cdots, w_L$。在回归中，我们取所有预测的加权平均。投票一词源于加权平均在分类中的使用。

$$y_i = \sum_{j=1}^{L} w_j d_{ji} \tag{15.3}$$

其中 d_{ji} 是学习器 j 对 C_i 类的投票，w_j 是其投票的权重。在最简单的情形下，我们有简单投票(simple voting)，其中所有投票者具有相同的权重，即 $w_j = 1/L$。在分类中，这称为简单多

数表决(plurality voting)，其中得票最多的类胜出。当只有两个类时，这就是(过半)多数表决(majority voting)，其中胜出类获取一半以上投票(习题1)。如果投票者还能提供它们为每个分类投票多少的额外信息(例如，通过后验概率)，则规范化后，这些信息即可用作加权投票方案的权重。同样，如果 d_{ji} 是类别后验概率 $P(C_i \mid x, \mathcal{M}_j)$，则我们可以简单将其相加($w_j = 1/L$)并选取具有最大 y_i 的类。

354 ≀ 355

另外一种可能性是在另外的确认集上评估学习器(回归或分类器)的准确率并使用这些信息来计算权重，使得可以对更准确的学习器赋予更高的权重。

投票方案可以看作贝叶斯框架下的近似，以权重近似先验模型概率，并以模型决策近似模型条件似然。这就是贝叶斯模型组合(Bayesian model combination)。例如，在分类中我们有 $w_j \equiv P(\mathcal{M}_j)$，$d_{ji} = P(C_i \mid x, \mathcal{M}_j)$，而(15.3)式相当于

$$P(C_i \mid x) = \sum_{\text{所有模型}\mathcal{M}_j} P(C_i \mid x, \mathcal{M}_j) P(\mathcal{M}_j) \tag{15.4}$$

简单投票相当于假定一致先验概率。如果一个先验分布更倾向于较简单的模型，则简单投票将赋予简单的模型更大的权重。我们可以不在所有模型上集成；我们只选取一个我们认为 $P(\mathcal{M}_j)$ 值高的子集，或者可以执行另一个贝叶斯步骤来计算给定样本上的模型的条件概率 $P(\mathcal{M}_j \mid \mathcal{X})$，并从该密度选取一些高概率的模型。

Hansen 和 Salamon(1990)证明：给定(预测)成功概率高于1/2(即比随机猜测的预测好)的一组独立的两类分类器，使用多数表决，预测准确率随着投票分类器个数的增加而提高。假设 d_j 服从独立同分布，其期望为 $E[d_j]$、方差为 $Var(d_j)$，那么当使用 $w_j = 1/L$ 取简单平均时，输出的期望和方差分别为

$$E[y] = E\left[\sum_j \frac{1}{L} d_j\right] = \frac{1}{L} L E[d_j] = E[d_j]$$

$$\text{Var}(y) = \text{Var}\left(\sum_j \frac{1}{L} d_j\right) = \frac{1}{L^2}\text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2} L \text{Var}(d_j) = \frac{1}{L}\text{Var}(d_j) \tag{15.5}$$

可以看到期望没有改变，因而偏倚也不改变。但是方差，进而均方误差随着独立投票者数量 L 的增加而降低。在一般情况下， 356

$$\text{Var}(y) = \frac{1}{L^2}\text{Var}\left(\sum_j d_j\right) = \frac{1}{L^2}\left[\sum_j \text{Var}(d_j) + 2\sum_j \sum_{i<j} \text{Cov}(d_j, d_i)\right] \tag{15.6}$$

由此，我们看到，如果投票者并非独立而是负相关的话，进一步降低方差是可能的。如果随此增加的偏倚不是更高的话，则误差也会降低。

如果将每个基学习器看作是附加在真实判别式/回归函数上的随机噪声函数，而且这些噪声函数是不相关的并且均值为0，那么在每个估计上平均就相当于在噪声上平均。从这种意义上讲，投票具有光滑函数空间的效果并可以看作是一个在真实函数上具有光滑假设的正则化子(Perrone1993)。我们在图4-5(d)看到一个例子，其中通过在具有大方差的模型上取平均，我们得到了比单独模型更好的拟合。这就是投票的思想：我们对具有高方差低偏倚的模型投票，使得在组合后，偏倚依然保持很小而通过取平均降低了方差。即使单个模型是有偏的，方差的降低也可能抵消偏倚并仍然可能降低误差。

15.3 纠错输出码

在纠错输出码(error-correcting output codes, ECOC)中(Dietterich 和 Bakiri 1995),主要的分类任务通过由基学习器实现的一组子任务来定义。其思想是:将一个类从其他类区分开来的原始任务可能是一个困难的问题。作为替代,我们可以定义一组简单的分类问题,每个专注于原始任务的一个方面,并通过组合这些简单分类器来得到最终的分类器。

这时,基学习器是输出为 $-1/+1$ 的二元分类器,并且有一个 $K \times L$ 的编码矩阵 $\mathbf{W}$,其 K 行是关于 L 个基学习器 d_j 的类的二元编码。例如,如果 $\mathbf{W}$ 的第二行是 $[-1, +1, +1, -1]$,则这意味如果一个实例属于 C_2 类,则该实例应在 d_1 和 d_4 上取负值,在 d_2 和 d_3 上取正值。类似地,编码矩阵的列定义了基学习器的任务。例如,如果第三列是 $[-1, +1, +1]^T$,可理解为第三个基学习器 d_3 的任务是将属于 C_1 类的实例与属于 C_2 和 C_3 类的实例区分开。这就是我们如何构成基学习器的训练集的方式。例如,在这个例子中,所有标记为 C_2 类及 C_3 类的实例形成 $\mathcal{X}_3^+$,而标记为 C_1 类的实例构成 $\mathcal{X}_3^-$,而对 d_3 的训练应使得当 $x^t \in \mathcal{X}_3^+$ 时输出 $+1$,当 $x^t \in \mathcal{X}_3^-$ 时输出 -1。

357

这样,编码矩阵使得我们可以用二分问题($K=2$ 的分类问题)定义多分问题($K>2$ 的分类问题),并且这是一种适用于任意可以实现二分基学习器的学习算法的方法,例如,线性或多层感知器(单输出)、决策树或初始定义用于两类问题的 SVM。

典型的每类一个判别式的情况对应于对角编码矩阵,其中 $L=K$。例如,对于 $K=4$,我们有

$$\mathbf{W} = \begin{bmatrix} +1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 \\ -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & +1 \end{bmatrix}$$

这里的问题是:如果某一个基学习器存在错误,就会有误分类,因为类的码字之间非常相似。因而纠错码采用的方法是使 $L>K$ 来增加码字之间的汉明距离。一种可能的方法是类逐对分开(pairwise separation),其中对 $i<j$ 有一个不同的基学习器将 C_i 与 C_j 分开(10.4节)。在这种情况下,当 $K=4$ 时 $L=K(K-1)/2$,编码矩阵为

$$\mathbf{W} = \begin{bmatrix} +1 & +1 & +1 & 0 & 0 & 0 \\ -1 & 0 & 0 & +1 & +1 & 0 \\ 0 & -1 & 0 & -1 & 0 & +1 \\ 0 & 0 & -1 & 0 & -1 & -1 \end{bmatrix}$$

其中的0表示“无关”。这就是说,训练 d_1 来将 C_1 与 C_2 分开并且在训练中不使用属于其他类的实例。类似地,一个实例属于 C_2 如果有 $d_1=-1$ 并且 $d_4=d_5=+1$,并且我们不考虑 d_2、d_3 和 d_6 的值。这种方法的问题是 L 是 $\mathcal{O}(K^2)$。因而,对于比较大的 K,逐对分开可能是不可行的。

方法是预先设置 L 值，然后寻找 $\mathbf{W}$ 使得以汉明距离衡量的行间距离以及列间距离都尽可能的大。对 K 个类而言，存在 $2^{(K-1)}-1$ 种可能的列，即两类问题。这是因为 K 位可写为 2^K 种不同的形式和补(比如，“0101”和“1010”，从我们的角度来看，二者定义相同的判别式)，将所有可能组合除以 2 然后减 1，因为全为 0(或 1)的列是无用的。例如，当 $K=4$ 时，我们有 358

$$\mathbf{W}=\begin{bmatrix}-1 & -1 & -1 & -1 & -1 & -1 & -1\\ -1 & -1 & -1 & +1 & +1 & +1 & +1\\ -1 & +1 & +1 & -1 & -1 & +1 & +1\\ +1 & -1 & +1 & -1 & +1 & -1 & +1\end{bmatrix}$$

当 K 很大时，对于一个给定的 L 值，我们从 $2^{(K-1)}-1$ 列选取 L 列。我们希望 $\mathbf{W}$ 的这些列尽可能的不相同，以便每个基学习器所学习的子任务尽可能互不相同。同时，我们希望 $\mathbf{W}$ 的行也尽可能不相同，使得在一个或多个基学习器失效时可以获得最大的纠错。

ECOC 可以投票方式来表述，其中 $\mathbf{W}$ 的元素 w_{ij} 被看作是投票权重：

$$y_i=\sum_{j=1}^{L} w_{ij}d_j \tag{15.7}$$

然后我们选取具有最高 y_i 的类。通过求加权和并选择最大值(判断类别)取代寻求一个精确的匹配使得 d_j 也不必是二元的，而是可取 -1 到 $+1$ 之间的任意值，以软确定性取代硬判决。注意，位于 0 和 1 之间的 p_j 值(例如后验概率)可以很简单地被转换为 -1 到 $+1$ 间的 d_j 值：

$$d_j=2p_j-1$$

(15.7)式与(15.3)式的一般投票模型的不同在于投票的权重对不同的类可以不同，即以 w_{ij} 取代了 w_j，并且 $w_j\geqslant 0$ 而 w_{ij} 为 -1、0 或 $+1$.

ECOC 的一个问题是：由于编码矩阵 $\mathbf{W}$ 被设置为先验，因此不能保证由 $\mathbf{W}$ 的列所定义的子任务一定是简单的。Dietterich 和 Bakiri(1995)的研究表明二分树可能要比多分树大，而且当使用多层感知器时，后向传播可能收敛较慢。 359

15.4 装袋

装袋(bagging)是一种投票方法，其中基学习器通过在稍有差异的训练集上训练而有所不同。从给定的样本数据上产生 L 个稍稍不同的样本集通过自助法来完成，其中给定一个大小为 N 的数据集 $\mathcal{X}$，随机从 $\mathcal{X}$ 中有放回地(with replacement)抽取 N 个实例(14.2.3 小节)。由于抽样是通过有放回方式完成的，可能某些实例被多次抽取而某些实例根本没有被抽到。当抽取 L 个样本 $\mathcal{X}_j(j=1,\cdots,L)$ 后，这些样本集是彼此相似的，因为它们是从相同的原始样本数据源抽取的，但是源于随机性而又稍有不同。基学习器 d_j 在这 L 个样本集的 $\mathcal{X}_j$ 上训练。一个学习算法是不稳定算法(unstable algorithm)，如果训练集中很小的变化会引起所产生学习器很大的差异，即学习算法具有高方差。装袋是自助聚集(bootstrap aggregating)的简单说法，就是使用自助法产生 L 个训练集，并使用不稳定的学习过程训练 L 个基学习器，并在检验时取(预测的)平均值(Breiman1996)。装袋可用于分类和回归。在用于回归的情况下，为

了更加鲁棒，可以在组合预测结果时以中值来取代平均值。

诸如决策树和多层感知器这样的算法是不稳定的。最近邻算法是稳定的，但是精简的最近邻算法是不稳定的(Alpaydın1997)。如果原始训练集很大，则我们可能希望使用自助法来从它产生小一些的数据集($N' < N$)，否则 $\mathcal{X}_j$ 的自助副本将会非常相似，从而 d_j 将高度相关。

15.5 提升

在装袋中，产生互补的基学习器是靠运气以及学习方法的不稳定性。在提升中，我们通过在前一个学习器所犯的错误上训练下一个学习器，积极地尝试产生互补的学习器。原始的提升(boosting)算法(Schapire1990)组合了三个弱学习器来产生一个强学习器。所谓弱学习器(weak learner)是误差概率小于 1/2 的学习器，这使得它对两类问题比随机猜测要好，而强学习器(strong learner)具有任意小的误差概率。

360 给定一个大训练集，我们随机地将其划分为三部分。使用 $\mathcal{X}_1$ 来训练 d_1。然后取 $\mathcal{X}_2$ 并将它馈入 d_1。将所有 d_1 误分类的实例以及在 $\mathcal{X}_2$ 中被 d_1 正确分类的一些实例一起作为 d_2 的训练集。然后取 $\mathcal{X}_3$ 并将它馈入 d_1 和 d_2。其中 d_1 和 d_2 输出不一致的实例形成 d_3 的训练集。在检验期间，给定一个实例我们首先将其提供给 d_1 和 d_2；如果二者输出一致，这就作为输出结果，否则 d_3 的输出作为结果。Schapire(1990)的研究表明这个整体系统降低了错误率，并且错误率可以通过递归地使用这样的系统(即将三个模型构成的提升系统作为更高层系统的 d_j)而任意降低。

尽管这种方法很成功，但是提升方法的不足之处在于需要一个非常大的训练集。样本需要一分为三，而且第二和第三个分类器只在由其前的分类器犯错的实例构成的子集上训练。因此，除非有一个很大的训练集，否则 d_2 和 d_3 将无法拥有合理大小的训练集。Drucker 等(1994)在其提出的提升多层感知器中使用了有 118000 个实例的数据集，用于光学手写数字识别。

Freund 和 Schapire(1996)提出了提升的一个变种，叫 AdaBoost，是自适应提升的缩写，其中重复使用相同的训练集因而不要求数据集很大。AdaBoost 还可以组合任意数量的基学习器，不一定是三个。

已经有很多 AdaBoost 的变种被提出；这里我们讨论原始的算法 AdaBoost. M1(见图 15-2)：其思想是将实例抽取的概率修改成误差的函数。令 p_j^t 表示实例对(x^t, r^t)被抽取用于训练第 j 个基学习器的概率。最初，所有的 $p_1^t = 1/N$。然后，以如下方式添加新的基学习器：从 $j = 1$ 开始，ε_j 表示 d_j 的错误率。AdaBoost 要求任意的 $\varepsilon_j < 1/2$；如果不满足，即停止添加新的基学习器。注意，这里的错误率并非基于原始问题，而是基于在第 j 步中使用的数据集。定义 $\beta_j = \varepsilon_j/(1-\varepsilon_j) < 1$，并且设置 $p_{j+1}^t = \beta_j p_j^t$ 如果 d_j 正确地对 x^t 分类，否则设置 $p_{j+1}^t = p_j^t$。由于 p_{j+1}^t 应该是概率，所以我们用 p_{j+1}^t 除以 $\sum_t p_{j+1}^t$ 对其规范化，使它们的和为 1。这样做的效果是将被正确分类的实例的(抽取)概率降低，而将被错误分类的实例的概率提高。然后，根据这些修改后的概率 p_{j+1}^t，从原样本中有放回地抽取相同大小的样本集，并用于训练 d_{j+1}。

训练：

For 所有的 $\{x^t, r^t\}_{t=1}^N \in \mathcal{X}$，初始化 $p_1^t = 1/N$

For 所有的基分类器 $j=1, \cdots, L$

按照概率 p_j^t 随机地从 $\mathcal{X}$ 抽取 $\mathcal{X}_j$

使用 $\mathcal{X}_j$ 训练 d_j

For 每个 (x^t, r^t)，计算 $y_j^t \leftarrow d_j(x^t)$

计算错误率：$\varepsilon_j \leftarrow \sum_t p_j^t \cdot 1(y_j^t \neq r^t)$

If $\varepsilon_j > 1/2$，then $L \leftarrow j-1$；Stop

$\beta_j \leftarrow \varepsilon_j/(1-\varepsilon_j)$

For 每个 (x^t, r^t) //如果正确，则减低概率

If $y_j^t = r^t$ then $p_{j+1}^t \leftarrow \beta_j p_j^t$ else $p_{j+1}^t \leftarrow p_j^t$

//规范化概率：

$Z_j \leftarrow \sum_t p_{j+1}^t$；$p_{j+1}^t \leftarrow p_{j+1}^t / Z_j$

检验：

给定 x，计算 $d_j(x)$，$j=1, \cdots, L$

计算类输出，$i=1, \cdots, K$：

$$y_i = \sum_{j=1}^{L} \left(\log \frac{1}{\beta_j} \right) d_{ji}(x)$$

图 15-2 AdaBoost 算法

这样做的效果是使得 d_{j+1}更专注于被 d_j 误分类的实例。这就是为什么基学习器以简单而不是准确为原则选取，否则下一个训练样本集将仅仅包含少数离群点和噪声实例的多次重复。例如，对于决策树，使用的是决策树桩(decision stump)，一种只有一层或两层的树。因此，很明显它们是有偏的但是方差上的降低比较大，而且总体误差也会降低。像线性判别式这样的算法本身具有低方差，我们不能通过 AdaBoost 而获得更低的方差。

一旦完成训练，AdaBoost 就采用投票方法。给定一个实例，所有的 d_j 决定其分类，而后取一个加权的投票结果，其中权重与基学习器(在训练集上的)准确率成正比：$w_j = \log(1/\beta_j)$。Freund 和 Schapire(1996)表明 AdaBoost 在 22 个基准问题上提高了准确率，在一个基准问题上准确率相同，而在 4 个基准问题上准确率较差。

Schapire 等(1998)认为 AdaBoost 的成功源于其扩展边缘(margin)。如果边缘增加，训练实例可以更好的被分隔而使误分类不易发生。这使得 AdaBoost 的目标和支撑向量机(10.9 节)类似。

在 AdaBoost 中，尽管不同的基学习器使用稍有差异的训练集，但是这种差异和装袋一样是靠运气，所不同的是它是前一个基学习器的误差的函数。提升针对一个特定问题的实际性能显然依赖于数据和基学习器。为此，需要有充足的训练数据，并且基学习器应当是弱的但又不是太弱，而且提升对噪声和离群点尤其敏感。

AdaBoost 已经被推广到回归：由 Avnimelech 和 Intrator(1997)提出的一种直截了当的方法是，检查预测误差是否大于某个阈值，如果是则将其标记为错误，然后使用正规的 AdaBoost。在另一个版本中(Drucker1997)，抽取概率根据误差量进行修改，使得前一个基

361
~
362

学习器预测误差较大的实例，在下一个基学习器的训练中有较大的概率被抽取。最后用加权平均或中值来组合这些基学习器的预测结果。

15.6 重温混合专家模型

在投票中，权重 w_j 在输入空间上是固定的。在混合专家模型构架中(12.8节)，存在一个门网络，其输出取作投票的权重。因而这一构架可以被看作是一种投票方法，其中投票依赖于输入，而且可能因输入不同而有所不同。混合专家模型使用的竞争学习算法局部化了基学习器，使得每个基学习器变成输入空间的一个不同领域的专家，并且其权重 $w_j(x)$ 在其专长的领域中接近于1。最终的输出与投票一样是加权平均

$$y = \sum_{j=1}^{L} w_j(x) d_j \tag{15.8}$$

不同之处在于基学习器和权重二者均是输入的一个函数(见图15-3)。

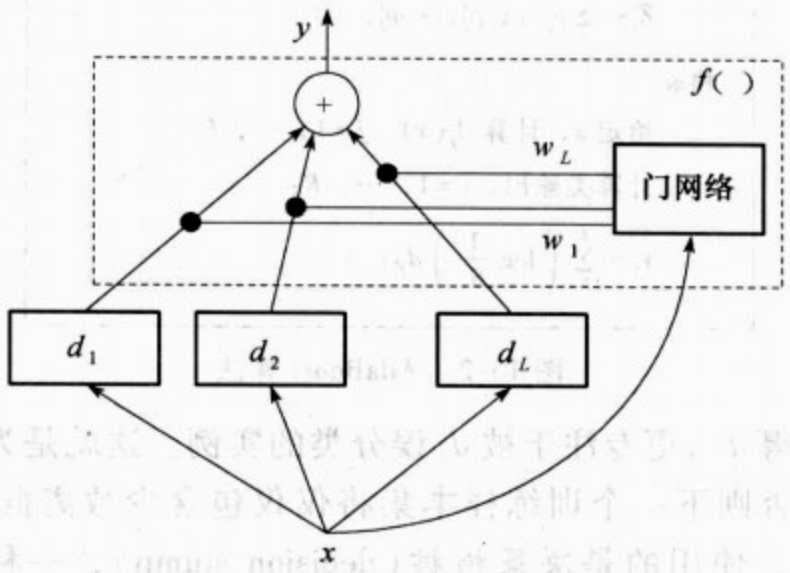


图15-3 混合专家模型是一种投票方法，其中，像由门网络给出的那样，投票是输入的函数。组合系统 f 也包含这种门系统

Jacobs(1997)显示在混合专家模型构架中，专家是有偏的，但是负相关的。随着训练的进行，偏倚降低而专家的方差增加，但与此同时，随着专家局部化于输入空间的不同部分，它们的协方差为负并且越来越小。根据(15.6)式，这将降低总体方差，进而降低误差。在12.8节，我们讨论了专家和门网络均为线性函数的情况，但是非线性方法同样可以用于二者。这将降低专家的偏倚，但是有增加专家方差和过分拟合的风险。

15.7 层叠泛化

层叠泛化(stacked generalization)是Wolpert(1992)提出的一种扩展的投票方法，其中基学习器的输出组合方式不必是线性的，而是通过一个组合器系统 $f(\cdot|\Phi)$。组合器是另一个学习器，其参数 Φ 也要训练(见图15-4)：

$$y = f(d_1, d_2, \cdots, d_L | \Phi) \tag{15.9}$$

当基学习器给出某种输出组合时，组合器学习什么是正确的输出。我们不能在训练数据上训练组合器函数，因为基学习器可能在记忆训练数据；组合器系统应当真正的学习基学习器是如何犯错(产生误差)的。因此组合器应当在基学习器训练时没有出现的数据上训练。

尽管开销很大，Wolpert 提出使用留一法，而当样本集很大时使用 k 折交叉确认更有效。

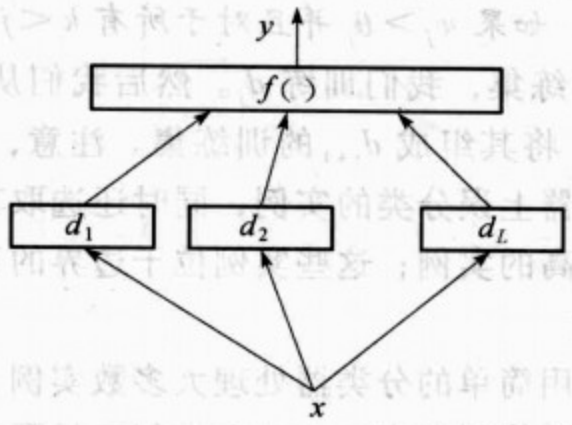


图 15-4　在层叠泛化中，组合器是另一个学习器，并且不必像投票一样是线性的

如果 $f(\cdot \mid w_1, \cdots, w_L)$ 是线性模型，其约束为 $w_i \geq 0$，$\sum_j w_j = 1$，则最佳权重可通过受约束的回归来获得。但是请注意，对组合器函数没有限制，并且不像投票，组合可以是非线性的。例如，$f(\cdot)$ 可以是一个多层感知器，Φ 是其连接权重。基学习器 d_j 的输出定义了一个新的 L 维空间，在该空间组合器函数学习输出的判别式/回归函数。

在层叠泛化中，我们希望基学习器尽可能不同，使得它们可以相互补充，并且每个基学习器都基于不同的学习算法是很可取的。Zhang、Mesirov 和 Waltz(1992)使用层叠进行蛋白质二级结构预测，显著提高了准确率。在他们的研究中，基学习器分别是参数分类器、最近邻分类器和一个多层感知器。组合器是另一个多层感知器。

15.8　级联

级联分类器的思想是使用一个基学习器 d_j 的序列，按照空间和时间复杂度或它们使用的数据表示的代价对其进行排序，使得 d_{j+1} 的代价高于 d_j(Kaynak 和 Alpaydın2000)。级联(cascading)是一种多级方法，并且只有在所有前驱学习器 d_k($k<j$)都不足够确信时才使用 d_j(见图 15-5)。为此，与每个学习器相关联的是一个置信度 w_j，当有 $w_j > \theta_j$ 时我们说 d_j 对其输出是确信的并且其结果可用，其中 $1/K < \theta_j \leq \theta_{j+1} < 1$ 是置信度阈值。在分类中，置信度函数被设置为最高的后验：$w_j \equiv \max_i d_{ji}$；这正是用于拒绝的策略(3.3 节)。

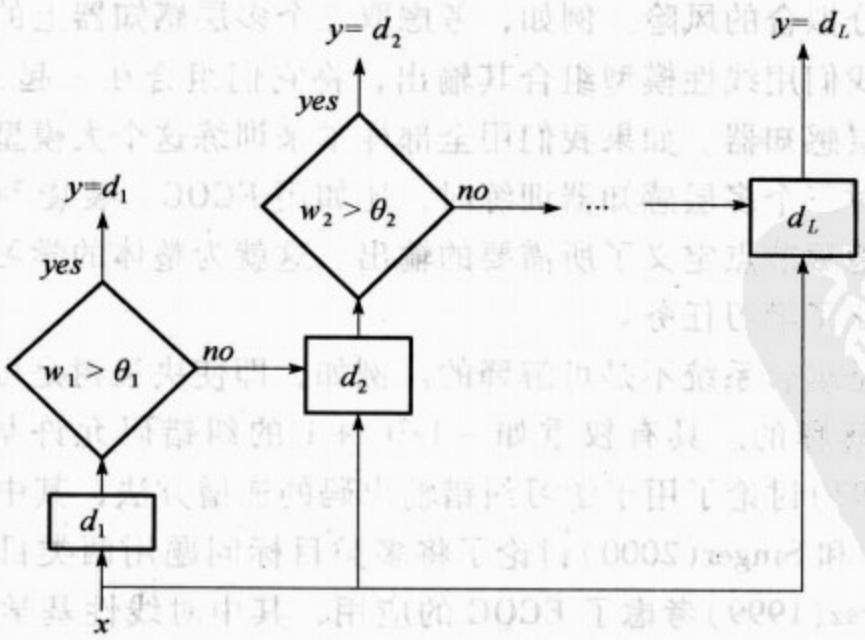


图 15-5　级联是一个多级方法，其中使用一个分类器序列，并且仅当前驱分类器不够好时才使用下一个分类器

如果所有前驱学习器的结果均不够确信，我们才使用学习器 d_j：

$$y_i = d_{ji} \quad \text{如果 } w_j > \theta_j \text{ 并且对于所有 } k < j \text{ 有 } w_k < \theta_k \tag{15.10}$$

从 $j=1$ 开始，给定一个训练集，我们训练 d_j。然后我们从另外一个确认集中找出所有使 d_j 不够好(不确信)的实例，将其组成 d_{j+1} 的训练集。注意，和 AdaBoost 不同的是，这里我们不仅选取在前一个基学习器上误分类的实例，同时还选取其不自信的实例。这包括误分类的实例以及后验概率不足够高的实例；这些实例位于边界的正确一侧，但是它们与判别式之间的距离(即边缘)不足够大。

级联的思想是：在初期使用简单的分类器处理大多数实例，而更为复杂的分类器仅用于少数实例，因此并不显著增加总体复杂度。这正和类似于投票的多专家方法相反，这些方法中，所有基学习器为每个实例产生输出。如果问题空间比较复杂，几个每一级的复杂性递增的基学习器可能级联。为了不增加基分类器的个数，少数没有被任何基分类器覆盖的实例将被原样保留，并通过一个非参数分类器(如 k-NN)来处理。

级联的归纳偏倚是类可以通过复杂度递增的少量“规则”来解释，并存在一个没有被这些规则覆盖的小的“异常”集合。这些规则通过简单的基学习器来实现，例如复杂度递增的感知器，学习在整个输入空间上有效的一般规则。异常是局部实例，最好由非参数模型处理。

因此，级联位于参数和非参数分类两个极端之间。前者(例如线性模型)寻找覆盖所有实例的单个规则。而非参数模型(如 k-NN)存储所有的实例而不产生任何解释它们的简单规则。级联产生一个(或多个)规则，以尽可能低代价地解释大部分实例，并将其余实例作为异常存储。这在很多学习应用中是有道理的。例如，在多数情况下，英语动词的过去式是在其后加“d”或“ed”；但也存在不规则动词并不符合这一规则的情况，例如“go/went”。

365
~
367

15.9 注释

组合学习器的思想是将复杂的任务划分为较简单的子任务，这些子任务可以由分别训练的基学习器处理。每个基学习器有其自己的子任务。如果我们用一个大的学习器包含所有的基学习器，则会有过分拟合的风险。例如，考虑取三个多层感知器上的投票，每个感知器具有一个隐藏层。如果我们用线性模型组合其输出，将它们组合在一起，则我们有一个大的、具有两个隐藏层的多层感知器。如果我们用全部样本来训练这个大模型，则很可能产生过分拟合。而当我们分别对三个多层感知器训练时，比如用 ECOC、装袋等，就如同为这个大的多层感知器的第二层隐藏节点定义了所需要的输出。这就为整体的学习器应对什么进行学习附加了约束，进而简化了学习任务。

组合的一个缺点是组合系统不是可解释的。例如，即使决策树是可解释的，装袋的或提升的决策树也不是可解释的。具有权重如 $-1/0/+1$ 的纠错码允许某种形式的可解释性。Mayoraz 和 Moreira(1997)讨论了用于学习纠错输出码的递增方法，其中基学习器在需要时添加。Allwein、Schapire 和 Singer(2000)讨论了将多类目标问题用两类目标问题编码的各种方法。Alpaydın 和 Mayoraz(1999)考虑了 ECOC 的应用，其中对线性基学习器组合得到非线性判别式，他们还提出了从数据中学习 ECOC 矩阵的方法。

最早也是最直观的方法就是投票。Xu、Krzyzak 和 Suen(1992)对早期工作进行了回顾。

Benedikttson 和 Swain(1992)使用投票方法进行多源组合。Kittler 等(1998)对近期的投票方法进行了回顾，同时讨论了对数据的多表示进行组合的应用。该人脸识别应用使用三种表示：正面人脸图像、人脸轮廓图像和声音。投票模型的误差率低于使用一种表示的误差率。Alimo ğlu 和 Alpaydın1997 给出了另一个针对手写数字识别的应用，其中对两种信息源进行了组合：一种是数字在触摸书写板书写时的笔移动时态数据，另一种是数字书写后的静态二维位图图像。在这个应用中，使用其中一种数据表示的两个分类器误差率约为 5%，但是通过组合误差率降至 3%。同时在该应用研究中还表明关键的阶段在于设计互补的学习器和/或数据表示，学习器的组合方式倒不像前者那样重要。 368

Jacobs(1995)表明 L 个依赖的专家和 L' 个独立的专家同样有价值，其中 $L' \leqslant L$。在某些环境下，投票模型与贝叶斯技术产生相同的结果(Jacobs1995)。(15.4)式的先验概率因而可以由超参数的分布来建模，并在理想情况下，应在整个模型参数空间进行集成。这种方法在实际中并非总是可行的，并且需要求助于近似或抽样。随着贝叶斯统计的进展，这些超贝叶斯技术可能会在不远的将来变得越来越重要。

自 20 世纪 90 年代初，组合多学习器就已经成为机器学习领域中的一个流行课题，从那时起研究一直在进行(Dietterich1997)。AdaBoost 目前被认为是最好的机器学习算法之一，并且一旦基学习器本身及其数量确定之后，它几乎是完全自动的。同时也存在 AdaBoost 的其他版本，其中下一个基学习器在前一个学习器的残余之上进行训练(Hastie、Tibshirani 和 Friedman2001)。在 www. boosting. org 网站上可以找到最近发表的关于模型组合的一般性以及专门针对 AdaBoost 的文献。尽管多模型(组合)的实际应用非常成功，但是关于模型组合如何或为什么有效的讨论仍然在继续；例如 Breiman1998、Bauer 和 Kohavi1999。

15.10 习题

1. 如果每个基学习器是独立同分布的并且正确的概率 $p > 1/2$，那么 L 个分类器上的一个多数表决给出正确答案的概率是什么？
2. 在装袋中，为了产生 L 个训练集，以 L 折交叉确认来替代自助法的效果如何？
3. 提出一个学习纠错输出码的增量算法，其中新的二类问题在需要时添加，以便更好地解决多类问题。
4. 使用线性感知器作为组合器函数的话，投票和层叠的区别是什么？
5. 在级联中，为什么要求 $\theta_{j+1} \geqslant \theta_j$？ 369

15.11 参考文献

Alimo ğlu, F., and E. Alpaydin. 1997. "Combining Multiple Representations and Classifiers for Pen-Based Handwritten Digit Recognition." In *Fourth International Conference on Document Analysis and Recognition*, 637 – 640. Los Alamitos, CA: IEEE Computer Society.

Allwein, E. L., R. E. Schapire, and Y. Singer. 2000. "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers." *Journal of Machine Learning Research* 1: 113 – 141.

Alpaydin, E. 1997. "Voting over Multiple Condensed Nearest Neighbors." *Artificial Intelligence Review* 11: 115 – 132.

Alpaydin, E., and E. Mayoraz. 1999. "Learning Error-Correcting Output Codes from Data." In *Ninth International Conference on Artificial Neural Networks*, 743 – 748. London: IEE Press.

Avnimelech, R., and N. Intrator. 1997. "Boosting Regression Estimators." *Neural Computation* 11: 499 – 520.

Bauer, E., and R. Kohavi. 1999. "An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants." *Machine Learning* 36: 105 – 142.

Benediktsson, J. A., and P. H. Swain. 1992. "Consensus Theoretic Classification Methods." *IEEE Transactions on Systems, Man, and Cybernetics* 22: 688 – 704.

Breiman, L. 1996. "Bagging Predictors." *Machine Learning* 26: 123 – 140.

Breiman, L. 1998. "Arcing Classifiers (with discussion)." *Annals of Statistics* 26: 801 – 849.

Dietterich, T. G. 1997. "Machine Learning Research: Four Current Directions." *AI Magazine* 18: 97 – 136.

Dietterich, T. G., and G. Bakiri. 1995. "Solving Multiclass Learning Problems via Error- Correcting Output Codes." *Journal of Artificial Intelligence Research* 2: 263 – 286.

Drucker, H., C. Cortes, L. D. Jackel, Y. Le Cun, and V. Vapnik. 1994. "Boosting and Other Ensemble Methods." *Neural Computation* 6: 1289 – 1301.

Drucker, H. 1997. "Improving Regressors using Boosting Techniques." In *Fourteenth International Conference on Machine Learning*, ed. D. H. Fisher, 107 – 115. San Mateo, CA: Morgan Kaufmann.

Freund, Y., and R. E. Schapire. 1996. "Experiments with a New Boosting Algorithm." In *Thirteenth International Conference on Machine Learning*, ed. L. Saitta, 148 – 156. San Mateo, CA: Morgan Kaufmann.

Hansen, L. K., and P. Salamon. 1990. "Neural Network Ensembles." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12: 993 – 1001.

Hastie, T., R. Tibshirani, and J. Friedman. 2001. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer.

Jacobs, R. A. 1995. "Methods for Combining Experts' Probability Assessments." *Neural Computation* 7: 867 – 888.

Jacobs, R. A. 1997. "Bias/Variance Analyses for Mixtures-of-Experts Architectures." *Neural Computation* 9: 369 – 383.

Kaynak, C., and E. Alpaydin. 2000. "MultiStage Cascading of Multiple Classifiers: One Man's Noise is Another Man's Data." In *Seventeenth International Conference on Machine Learning*, ed. P. Langley, 455 – 462. San Francisco: Morgan Kaufmann.

Kittler, J., M. Hatef, R. P. W. Duin, and J. Matas. 1998. "On Combining Classifiers." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20: 226 – 239.

Mayoraz, E., and M. Moreira. 1997. "On the Decomposition of Polychotomies into Dichotomies." In *Fourteenth International Conference on Machine Learning* ed. D. H. Fisher, 219 – 226. San Mateo, CA: Morgan Kaufmann.

Perrone, M. P. 1993. "Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure." Ph. D. thesis, Brown University.

Schapire, R. E. 1990. "The Strength of Weak Learnability." *Machine Learning* 5: 197 – 227.

Schapire, R. E., Y. Freund, P. Bartlett, and W. S. Lee. 1998. "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods." *Annals of Statistics* 26: 1651 – 1686.

Wolpert, D. H. 1992. "Stacked Generalization." *Neural Networks* 5: 241 – 259.

Xu, L., A. Krzyzak, and C. Y. Suen. 1992. "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition." *IEEE Transactions on Systems, Man, and Cybernetics* 22: 418 – 435.

Zhang, X., J. P. Mesirov, and D. L. Waltz. 1992. "Hybrid System for Protein Secondary Structure Prediction." *Journal of Molecular Biology* 225: 1049 – 1063.

第16章 增强学习

在增强学习中，学习器是一个制定决策的智能主体。智能主体在其所处的环境中执行一些动作并根据其试图解决一个问题所执行的动作而获得奖励(或惩罚)。经过反复尝试运行，学习程序应当可以学习得到最优策略，即一个最大化总体奖励的动作序列。

16.1 引言

假设我们要构建一个学习下国际象棋的机器。在这种情况下，我们不能使用监督学习，原因有二：首先，请一位国际象棋老师带领我们遍历许多棋局并告诉我们每个位置的最佳棋步的代价非常昂贵。其次，在很多情况下，根本就没有最佳棋步；一个棋步的好坏依赖于其后的多个棋步。单一的棋步并不算数；而如果经过一个棋步序列我们赢得了比赛，则该棋步序列才是好的。而整个过程唯一的反馈是在最后我们赢得或是输掉棋局时才产生。

另一个例子是置于迷宫中的机器人。机器人按照四个罗盘方向之一进行移动，并进行一系列的移动到达迷宫出口。只要机器人在迷宫中，就不存在反馈，并且机器人尝试各种移动，直至到达出口，只有这时它才得到一个奖励。在这种情况下，机器人不存在对手，但是我们可能更偏好更短的(到达出口)的路径，这意味着我们是在和时间比赛。

这两个应用有多个共同点：存在一个称之为智能主体(agent)的决策者，并置其于某一环境(environment)中(见图 16-1)。在国际象棋的例子中，棋手是决策者而环境是棋盘；在第二个例子中，迷宫是机器人的环境。在任何时候，环境总是处于某种状态(state)，该状态来自于一组可能的状态之一，例如，棋盘的布局状态，机器人在迷宫中的位置。决策者可以做一组可能的动作：棋盘上棋子的合法移动，机器人沿着可能的方向移动而不会撞墙等。一旦选择并做了某一动作，状态就随之改变。问题的解决需要执行一系列的动作，之后我们才得到反馈，反馈以极少发生的奖励(reward)的形式给出，通常只有在完整的动作序列执行完毕才发生。奖励对问题进行了定义，同时是构建一个会学习的(learning)智能主体所必须的。会学习的智能主体学习解决一个问题的最佳步骤，其中“最佳”是以获取最大累积奖励的动作序列来量化。以上就是增强学习(reinforcement learning)的背景。

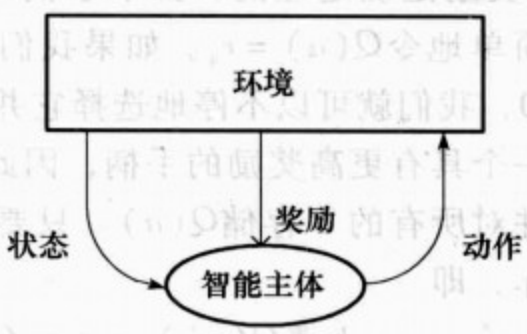


图 16-1 智能主体和环境进行交互。在环境的任意一个状态，智能主体执行一个改变环境状态的动作并获得一个奖励

增强学习与之前讨论的各种学习方法的不同之处在于以下几个方面：它称之为“和批评者一起学习”，而与之前和老师一起学习的监督学习方法相反。批评者(critic)不同于老师之处在于他并不告诉我们做什么，而仅仅告诉我们之前所做的怎么样；批评者永远不会提前提供信息。批评者提供的反馈极少，并且当他提供时，也是事后提供。这就导致了信度分配(credit assignment)问题：在执行若干动作并获得奖励后，我们希望对之前所执行的单个动作进行评估并找到可以引领我们赢得奖励的那些动作，以便对其记录并在之后使用。正如我们即将看到的，一个增强学习程序所做的是为中间状态或动作产生一个内部值(internal value)，来
374 表明这些状态或动作在引领我们达到目标并获取真正的奖励方面有多好。一旦学习到这样的内部奖励机制，智能主体就可以只执行最大化内部奖励的局部的动作。

问题的解决需要执行一个动作序列，而从这一角度，我们可想到第13章讨论的马尔可夫模型。事实上，我们使用马尔可夫决策过程来对智能主体建模。不同之处在于，对于马尔可夫模型，存在一个外部过程来产生一个我们对其观测和建模的信号序列，如语音。而在增强学习中，产生动作序列的是智能主体。之前，我们还区别可观测的和隐藏的马尔可夫模型，分别对应系统状态是可观测的或是隐藏的(并且也需要推断)。类似地，有时候我们使用一个部分可观测的马尔可夫决策过程来建模，其中智能主体不是确切地知道其所处的状态，而是需要通过使用传感器的观测以某种非确定性对其进行推断。例如，机器人在房间中移动时，机器人可能不知道其在房间中的确切位置，也不知道障碍物和目标的确切位置，而是通过一个照相机提供的有限图像来做决策。

16.2 单状态情况：K臂赌博机问题

我们从一个简单的例子开始。K臂赌博机是一种假想的具有K个手柄的老虎机。可做的动作是选择并拉下其中的一个手柄，而由此所赢取的一定量的钱就是和这个手柄(动作)相关联的奖励。任务是决定拉下哪个手柄，以便得到最大奖励。这是一个分类问题，其中我们选择K个手柄中的一个。如果是监督学习，则老师会告诉我们正确的类，即产生最大收益的类。而在增强学习中，我们只能尝试不同的手柄并记录其中最好的。这是一个简化的增强学习问题，因为只有一个状态，或者说只有一个老虎机，而我们只需要确定所执行的动作。另一个称其为简化问题的原因是我们在一个动作之后立即得到一个奖励；奖励并没有被延迟，因此在动作之后可以立即看到其价值。

假设$Q(a)$是动作a的价值。最初，对所有a都有$Q(a)=0$。当我们尝试执行动作a时，我们获得一个奖励$r_a \geqslant 0$。如果奖励是确定性的，拉下手柄a我们总是获得相同的奖励r_a，并且在这种情况下，我们可以简单地令$Q(a)=r_a$。如果我们想充分利用已有发现，一旦我们发现一个动作a具有$Q(a)>0$，我们就可以不停地选择它并在每次拉下手柄的时获得r_a的
375 奖励。然而，很可能还存在另一个具有更高奖励的手柄，因此我们还需要进行探索。

我们可以选择不同的动作并对所有的a存储$Q(a)$。只要我们想充分利用已有发现，我们可以选择具有最高价值的动作，即

$$\text{选择 } a^*, \quad \text{如果} Q(a^*) = \max_a Q(a) \tag{16.1}$$

如果奖励并非确定的而是随机的，则在选择相同的动作时我们每次获得不同的奖励。奖励量被概率分布$p(r \mid a)$定义。在这种情况下，我们定义$Q_t(a)$作为在时刻t时执行动作a的

价值的估计。它是在时刻 t 之前所有执行动作 a 所获奖励的平均值。一种在线更新方法可定义如下：

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta[r_{t+1}(a) - Q_t - (a)] \tag{16.2}$$

其中 $r_{t+1}(a)$ 是在 $(t+1)$ 时刻执行动作 a 之后所获得的奖励。

注意(16.2)式正是我们在之前章节中多次使用的 delta 规则(delta rule)：η 是学习因子(为了收敛而随时间逐渐降低)，r_{t+1} 是期望输出，而 $Q_t(a)$ 是当前的预测。$Q_{t+1}(a)$ 是在 $t+1$ 时刻的动作 a 的期望值，并且随着 t 的增加收敛到 $p(r \mid a)$ 的均值。

完整的增强学习问题从以下几个方面推广了这种简单情况：首先，我们有多个状态。这相当于同时存在具有多个不同奖励概率 $p(r \mid s_i, a_j)$ 的老虎机，而我们需要对 $Q(s_i, a_j)$，即在状态 s_i 时执行动作 a_j 的价值进行学习。其次，动作不仅影响获得的奖励而且影响下一状态，并且我们从一个状态转移到另一个状态。第三，奖励被延迟，而我们需要能够从延迟的奖励值估计立即值。

16.3 增强学习基础

学习的决策者称为智能主体(agent)。智能主体与环境(environment)之间进行交互。环境包含了除智能主体之外的所有东西。智能主体具有感知器，用来决定其所处的状态(state)。当智能主体执行一个动作时，环境提供一个奖励(reward)。时间被离散化为 $t=0, 1, 2, \ldots$，并且 $s_t \in S$ 表示智能主体在时刻 t 的状态，其中 S 是所有可能的状态集合。$a_t \in \mathcal{A}(s_t)$ 表示智能主体在时刻 t 所执行的动作，其中 $\mathcal{A}(s_t)$ 是在状态 s_t 时所有可能执行的动作集合。当处于状态 s_t 的智能主体执行动作 a_t 时，时钟嘀嗒，接收到奖励 $r_{t+1} \in \Re$，并且智能主体转移到下一个状态 s_{t+1}。整个问题通过马尔可夫决策过程(Markov decision process，MDP)来建模。奖励和下一状态分别采样于它们相应的概率分布 $p(r_{t+1} \mid s_t, a_t)$ 和 $P(s_{t+1} \mid s_t, a_t)$。注意，我们所具有的是一个马尔可夫(Markov)系统，其中下一时刻的状态和奖励仅仅依赖于当前状态和动作。在一些应用中，奖励和下一状态是确定的，并且对某个状态和所执行的动作，存在一个可能的奖励值和下一状态。

376

依赖于应用，某一状态可能被指定为初始状态，而在一些应用中，也存在一个停止搜索的吸收终止(目标)状态；所有在终止状态执行的动作都以概率 1 将状态转移到自身并且没有任何奖励。从初始状态到终止状态的动作序列称为一个片段(episode)，或一次试验(trial)。

策略(policy)π 定义了智能主体的行为并且是从环境状态到动作之间的一个映射：$\pi: S \rightarrow \mathcal{A}$。策略定义了在任意状态 s_t 可以执行的动作：$a_t = \pi(s_t)$。策略 π 的价值 $V^\pi(s_t)$ 是从状态 s_t 开始，遵循该策略的智能主体所获得的期望累积奖励。

在有限阶段(finite-horizon)或片段(episodic)模型中，智能主体试图最大化下 T 个步骤的期望奖励：

$$V^\pi(s_t) = E[r_{t+1} + r_{t+2} + \cdots + r_{t+T}] = E\left[\sum_{i=1}^{T} r_{t+i}\right] \tag{16.3}$$

某些任务是连续的，并且不存在预先固定的关于片段的限制。在无限阶段(infinite-horizon)模型中，不存在序列长度的限制，但是未来的奖励将被打折扣：

$$V^\pi(s_t) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots] = E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}\right] \tag{16.4}$$

其中 $0 \leqslant \gamma < 1$ 是折扣率(discount rate)，保证所返回的奖励是有限的。如果 $\gamma = 0$，则只有立即的奖励算数。随着 γ 趋向于 1，处于未来的奖励将更多的被计算在内，而这时我们说智能主体变得更有远见了。γ 是小于 1 的，因为对于解决问题的动作序列总是会有一个时间上的限制，而且智能主体也许是一个靠电池运作的机器人。因此我们喜欢更早获得奖励而非更
377 晚，因为我们不确定智能主体可以运行多久。

对每个策略 π，存在其价值 $V^{\pi}(s_t)$，我们想要找到最优策略(optimal policy) π^* 使得

$$V^*(s_t) = \max_{\pi} V^{\pi}(s_t), \forall s_t \tag{16.5}$$

在某些应用中，例如在控制中，我们更希望处理成对的状态-动作值 $Q(s_t, a_t)$，而不是简单的状态值 $V(s_t)$。$V(s_t)$ 表示智能主体处于状态 s_t 的价值，而 $Q(s_t, a_t)$ 表示当处于状态 s_t 时执行动作 a_t 的价值。我们定义 $Q^*(s_t, a_t)$ 为处于 s_t 状态时执行动作 a_t 并在其后遵循最优策略的期望累积奖励。状态的价值等于其上可采取的最优动作的价值：

$$\begin{aligned} V^*(s_t) &= \max_{a_t} Q^*(s_t, a_t) \\ &= \max_{a_t} E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}\right] \\ &= \max_{a_t} E\left[r_{t+1} + \gamma \sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i+1}\right] \\ &= \max_{a_t} E[r_{t+1} + \gamma V^*(s_{t+1})] \\ V^*(s_t) &= \max_{a_t} \left(E[r_{t+1}] + \gamma \sum_{s_{t+1}} p(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})\right) \end{aligned} \tag{16.6}$$

对于每一个可能的下一状态 s_{t+1}，我们以概率 $P(s_{t+1} \mid s_t, a_t)$ 转移到 s_{t+1} 并自此遵循最优策略，所得的期望累积奖励是 $V^*(s_{t+1})$。我们在所有可能的下一状态上求和，并且打折扣，因为它晚一个时间步。加上立即期望奖励，我们得到动作 a_t 的总体期望累积奖励。最后我们选择所有动作中最好的一个。(16.6)式称为 Bellman 公式(Bellman's equation)(Bellman 1957)。类似地，我们还可以有

$$Q^*(s_t, a_t) = E[r_{t+1}] + \gamma \sum_{s_{t+1}} p(s_{t+1} \mid s_t, a_t) \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \tag{16.7}$$

一旦获得了 $Q^*(s_t, a_t)$ 的值，我们就可以定义策略 π 为执行动作 a_t^*，它在所有 $Q^*(s_t, a_t)$ 中具有最大值：

378
$$\pi^*(s_t): \text{选择 } a_t^*, \text{其中} Q^*(s_t, a_t^*) = \max_{a_t} Q^*(s_t, a_t) \tag{16.8}$$

这意味着只要我们获得所有 $Q^*(s_t, a_t)$ 的值，那么在每个局部步骤中使用贪心搜索，我们就可以得到一个最优的步骤序列，该序列最大化累积(cumulative)奖励。

16.4 基于模型的学习

我们从基于模型的学习开始，其中我们完全知晓环境模型的参数 $p(r_{t+1} \mid s_t, a_t)$ 和 $P(s_{t+1} \mid s_t, a_t)$。在这种情况下，我们不要进行任何探索就可以使用动态规划直接对最优价值函数和策略求解。最优价值函数是唯一的，即为 (16.6) 式的解。一旦获得了最优价值函数，最优策略即为选择最大化下一状态价值的动作：

$$\pi^*(s_t) = \arg\max_{a_t}\left(E[r_{t+1} \mid s_t, a_t] + \gamma \sum_{s_{t+1} \in s} P(s_{t+1} \mid s_t, a_t) V^*(s_t)\right) \tag{16.9}$$

16.4.1 价值迭代

为了找到最优策略，可以使用最优价值函数，并且存在一个称为价值迭代(value iteration)的迭代算法，业已证明它收敛于正确的 V^* 值。价值迭代算法的伪代码在图 16-2 中。

将 $V(s)$ 初始化为任意值
Repeat
　　For 所有的 $s \in S$
　　For 所有的 $a \in \mathcal{A}$
　　$Q(s, a) \leftarrow E[r \mid s, a] + \gamma \sum_{s' \in s} P(s' \mid s, a) V(s')$
　　$V(s) \leftarrow \max_a Q(s, a)$
Until $V(s)$ 收敛

图 16-2　基于模型学习的价值迭代算法

我们说价值迭代是收敛的，条件为两次迭代之间的最大价值差小于某个阈值 δ：

$$\max_{s \in S} \left| V^{(l+1)}(s) - V^{(l)}(s) \right| < \delta$$

其中l是迭代计数。由于我们只关心具有最大价值的动作，因此有可能在价值收敛于最优价值之前策略就收敛于最优策略。每次迭代的复杂度是$\mathcal{O}(|S|^2|\mathcal{A}|)$，但是下一个可能状态数目 $k < |S|$ 很小，因此复杂度降低到$\mathcal{O}(k|S||\mathcal{A}|)$。

16.4.2 策略迭代

在策略迭代中，我们直接存储和更新策略，而非间接地通过价值迭代寻求最优策略。图 16-3 给出了其伪代码。其思想是从一个策略开始，不断的改进它直到没有改变为止。价值函数可通过求解线性方程来计算。然后检验是否可以通过将这些解考虑在内而改进策略。这一步骤保证了对策略的改进，并且当不再可能继续改进时，可以确保所得策略是最优的。该算法每次迭代的时间复杂度是$\mathcal{O}(|\mathcal{A}||S|^2 + |S|^3)$，比价值迭代的复杂度高，但是策略迭代比价值迭代需要更少的迭代次数。

任意初始化一个策略 π'
Repeat
$\pi \leftarrow \pi'$
　　通过解线性方程组，计算使用 π 的价值
　　$V^{\pi}(s) = E[r \mid s, \pi(s)] + \gamma \sum_{s' \in s} P(s' \mid s, \pi(s)) V^{\pi}(s')$
　　在每个状态上改进策略
　　$\pi'(s) \leftarrow \text{argmax}_a (E[r \mid s, a] + \gamma \sum_{s' \in s} P(s' \mid s, a) V^{\pi}(s'))$
Until $\pi = \pi'$

图 16-3　基于模型学习的策略迭代算法

16.5 时间差分学习

模型通过奖励和下一状态概率分布来定义，而且从16.4节可以看到，当这些值均已知时，可以使用动态规划来求解最佳策略。然而，这些方法代价很高，并且我们很少具有如此完全的关于环境的知识。增强学习更有趣和更实际的应用是当我们并不知道模型的时候。这时，我们需要对环境进行探索来查询模型。我们首先讨论如何进行探索，而后我们讨论在确定和非确定情况下的无模型学习算法。尽管我们并不假定关于环境模型的全部知识是已知的，但是还是要求模型是固定的。

379
~
380

像我们稍后将要看到的，当我们进行探索并得以看到下一个状态的价值和奖励时，我们利用这一信息来更新当前状态的价值。这些算法称为*时间差分*(temporal difference)算法，因为我们所做的是考察一个状态(或状态-动作对)的价值的当前估计值与下一状态和所得到奖励的折扣值之间的差。

16.5.1 探索策略

为了对环境进行探索，一种可能性是使用ε-贪心(greedy)搜索，其中我们以概率ε在所有可能的动作中均匀、随机地选择一个动作，即进行探索；而以概率$1-\varepsilon$选择已知的最好动作，即进行利用。我们并不想无限地持续探索，而是一旦进行了足够的探索就开始对其利用；为此，我们以一个较大的ε值开始，并逐渐的减小它。我们需要确认所采取的策略是软(soft)策略，也就是说，在状态$s \in S$执行任意动作$a \in \mathcal{A}$的概率大于0。

我们可以根据概率进行选择，使用软最大函数将价值转化为概率

$$P(a \mid s) = \frac{\exp Q(s,a)}{\sum_{b=1}^{\mathcal{A}} \exp Q(s,b)} \tag{16.10}$$

然后根据这些概率对动作进行选择。为了逐渐地从探索向利用进行转移，我们可以使用一个“温度”变量T，并定义选择动作a的概率为

$$P(a \mid s) = \frac{\exp[Q(s,a)/T]}{\sum_{b=1}^{\mathcal{A}} \exp[Q(s,b)/T]} \tag{16.11}$$

当T很大的时候，所有的概率相等，因而我们进行的是探索。而当T很小的时候，更好的动作将受青睐。因此这时的策略是以一个大的T值开始并逐渐的减小它，这称为*退火*
381 (annealing)过程，在这种情况下就是在时间上从探索平滑地过渡到利用。

16.5.2 确定性奖励和动作

在无模型学习中，我们首先讨论较为简单的确定性情况，其中对任意一对状态-动作，只有一个奖励和可能的下一状态。在这种情况下，(16.7)式简化为

$$Q(s_t,a_t) = r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1},a_{t+1}) \tag{16.12}$$

而我们简单地将其作为一个赋值来更新$Q(s_t, a_t)$。当在状态s_t时，我们使用之前所见到的各种随机策略之一选择一个动作a_t，其返回一个奖励r_{t+1}并将状态转移到s_{t+1}。然后，前一动作的价值更新为

$$\hat{Q}(s_t, a_t) \leftarrow r_{t+1} + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}) \tag{16.13}$$

其中Q之上加帽表示该值为估计值。$\hat{Q}(s_{t+1}, a_{t+1})$是一个稍晚的值，因此有更高的概率是正确的。我们以$\gamma$对其进行折扣并加上立即奖励(如果有的话)，并将此作为前一个$\hat{Q}(s_t, a_t)$的新估计。这称为后推(backup)，因为这可以被看作是在下一个时间步骤中对一个动作的价值进行估计，并“将其后退”用来修改一个当前动作的价值估计。

目前，我们假定所有的值$\hat{Q}(s_t, a_t)$存储于一张表中；稍后，我们会讨论当$|S|$和$|\mathcal{A}|$很大时如何更为简洁地存储这些信息。

最初所有$\hat{Q}(s_t, a_t)$都为0，并且作为试验片段的结果及时更新。假设我们有一个状态转移的序列，并且在每次转移中，我们都使用(16.13)式，用当前状态-动作对的Q值来更新前一对状态-动作的Q值的估计。在中间状态，所有的奖励为0从而价值为0，因此不进行更新。当到达目标状态的时候，我们得到奖励r，因而可以将前一对状态-动作的Q值更新为γr。对于这个之前的状态-动作对，由于其立即奖励为0而来自下一对状态-动作对的贡献又因为晚一步而以γ折扣。于是，在下一个片段中，如果我们再次到达这个状态，我们将其前一状态更新为$\gamma^2 r$，以此类推。按照这种方式，经过多个片段之后，这一信息被后推到更早的状态-动作对。随着我们寻找到具有更高累积奖励的路径，如更短的路径，Q值不断递增直到最优值，而且这些Q值绝不会降低(参见图16-4)。

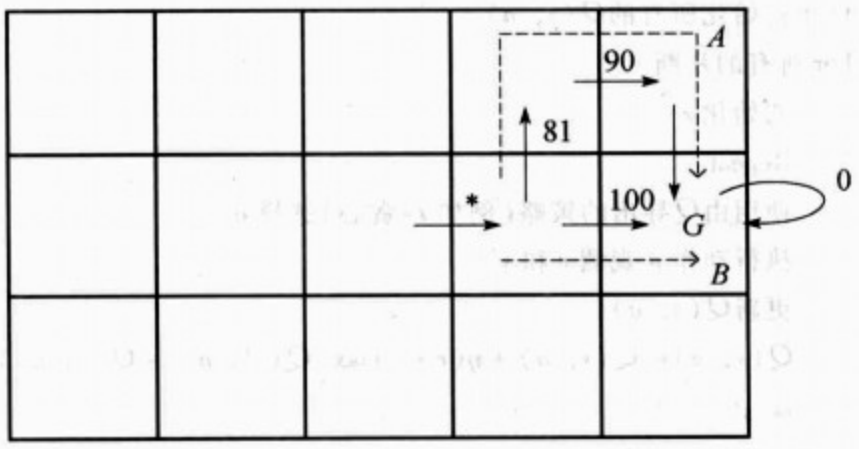


图16-4 说明Q值只增不减的例子。图示是一个确定的网格世界，其中G是目标状态并具有奖励100，所有其他立即奖励为0并有$\gamma = 0.9$。考虑由星号标记的转移的Q值，而且只考虑A和B两条路径。假设在看到路径B之前先看到路径A，则有$\gamma\max(0, 81) = 72.9$。如果之后又看到了B，则找到了更短的路径，而Q的值变为$\gamma\max(100, 81) = 90$。如果路径B在A之前被看到，则Q值为$\gamma\max(100, 0) = 90$。于是，当看到B时，(寻找到路径A时)Q的值不变，因为$\gamma\max(100, 81) = 90$

注意，这里我们并不知道奖励或下一状态函数。这些是环境的一部分，就好像是我们在探索的时候对其进行查询。我们也不对其进行建模，虽然有此可能性。我们只是原样接受它们并通过估计的价值函数来直接学习最优策略。

16.5.3 非确定性奖励和动作

如果奖励和动作的结果不是确定性的，则我们有一个奖励从中抽样的概率分布 $p(r_{t+1} \mid s_t, a_t)$，并且存在一个下一状态所服从的概率分布 $P(s_{t+1} \mid s_t, a_t)$。这些概率分布函数帮助我们对环境中不可控制的力量所引发的非确定性进行建模。这些不可控制的力量如国际象棋中的对手，西洋双陆棋中的骰子，或者是我们对系统知识的匮乏。例如，或许我们有一个不完美的机器人，它有时候会无法按预定的方向前进而产生偏离，或者比期望的距离走得更近或更远。

在这种情况下，我们有

$$Q(s_t, a_t) = E[r_{t+1}] + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a_t) \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{16.14}$$

382
~
383

在这种情况下，我们不能进行直接的赋值，因为对于相同的状态或动作，我们可能获得不同的奖励或者转移到不同的下一状态。我们所做的是取移动平均。这称之为Q学习(Q learning)算法：

$$\hat{Q}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \eta(r_{t+1} + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{16.15}$$

我们将这些 $r_{t+1} + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1})$ 值看作每个 (s_t, a_t) 对的实例的一个样本，并希望 $\hat{Q}(s_t, a_t)$ 收敛到其均值。与通常一样，为了收敛，η 的值随时间递减，并且已经证明该算法收敛于最优的 Q^* 值(Watkins 和 Dayan1992)。Q学习算法的伪代码见图 16-5。

```
任意初始化所有的Q(s, a)
For 所有的片断
    初始化 s
    Repeat
    使用由Q导出的策略(例如 ε-贪心)选择 a
    执行动作 a 观测 r 和 s'
    更新Q(s, a):
    Q(s, a)←Q(s, a) + η(r + γ max_a' Q(s', a') - Q(s, a))
    s←s'
Until s 是终止状态
```

图 16-5 Q学习，它是一种离策略时间差分学习算法

我们还可以认为(16.15)式的作用是减小当前的Q值和一个时间步骤之后的被后推的估计之间的差。这类算法称之为时间差分(temporal difference，TD)算法(Sutton1988)。

这是一种离策略(off-policy)方法，因为该方法使用下一个最优动作的值是而不使用策略。在一个在策略(on-policy)方法中，策略还用于确定下一个动作。Q学习的在策略版本就是 Sarsa 算法，其伪代码见图 16-6。我们看到，在策略的 Sarsa 算法使用从Q值推演出的策略来选择下一个动作 a'，并使用该动作的Q值来计算时间差分，而不是寻找所有可能的下一动作 a' 并选择其中最好的。在策略方法估计一个策略的价值并用它来执行动作。而在离策略
384 方法中，这些部分都是分离的，并且用于产生行为的策略称为行为(behavior)策略。行为策略事实上可能不同于称为估计(estimation)策略的被评估和被改进的策略。

如果采用 GLIE 策略来选择动作，Sarsa 算法以概率 1 收敛到最优策略和状态-动作值。GLIE(Greedy in Limit with Infinite Exploration，使用无限探索的极限贪心)策略是(1)所有状态-动作对都被无限次访问，并且(2)策略收敛到贪心策略的极限(贪心策略是可设定的，比如，使用 ε-贪心策略时设定 $\varepsilon = 1/t$)。

除了$Q(s, a)$之外，时间差分相同的思想还可以用于学习 $V(s)$值。TD 学习(*TD* learning)(Sutton1988)使用如下的更新规则来更新一个状态值：

$$V(s_t) \leftarrow V(s_t) + \eta[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{16.16}$$

上式依然是一个 delta 规则，其中 $r_{t+1} + \gamma V(s_{t+1})$是更好的、后一时刻的预测，而 $V(s_t)$是当前的预测。它们之间的差即为时间差分，而更新是为了减小这个差。更新因子 η 逐渐减小，因而 TD 确保收敛到最优值函数 $V^*(s)$。

任意初始化所有的$Q(s, a)$
For 所有的片段
    初始化 s
    使用由Q导出的策略(例如 ε-贪心)选择 a
    Repeat
    执行动作 a 观测 r 和 s'
    使用由Q导出的策略(例如 ε-贪心)选择 a'
    更新$Q(s, a)$:
    $Q(s, a) \leftarrow Q(s, a) + \eta(r + \gamma Q(s', a') - Q(s, a))$
    $s \leftarrow s'$, $a \leftarrow a'$
Until s 是终止状态

图 16-6　Sarsa 算法，它是Q学习算法的在策略版本

16.5.4　资格迹

前述算法均为单步算法，因为时间差分仅用于更新前一个(状态值或状态-动作对的)值。资格迹(eligibility traces)是对以往出现的状态-动作对的一个记录，它使得我们可以实现 385
时间信度分配，并且还可以更新以往达到的状态-动作对的值。我们以 Sarsa 算法学习Q值为例来说明这些都是如何完成的。对其进行修改来学习 V 值是直截了当的。

为了存储资格迹，需要为每个状态-动作对关联一个附加的内存变量 $e(s, a)$，初始化为 0。当状态-动作对(s, a)被访问时，也就是说在状态 s 执行了动作 a 时，其资格被设置为 1；其他所有状态-动作对的资格乘以 $\gamma\lambda$。$0 \leqslant \lambda \leqslant 1$ 是迹衰减参数。

$$e_t(s,a) = \begin{cases} 1 & \text{如果 } s = s_t \text{ 并且 } a = a_t, \\ \gamma\lambda e_{t-1}(s,a) & \text{否则} \end{cases} \tag{16.17}$$

如果某一状态-动作对从未被访问过，则其资格保持为 0；如果被访问过，随着时间流逝和其他状态-动作对被访问，该状态的资格依赖于 γ 和 λ 的值进行衰减(见图 16-7)。

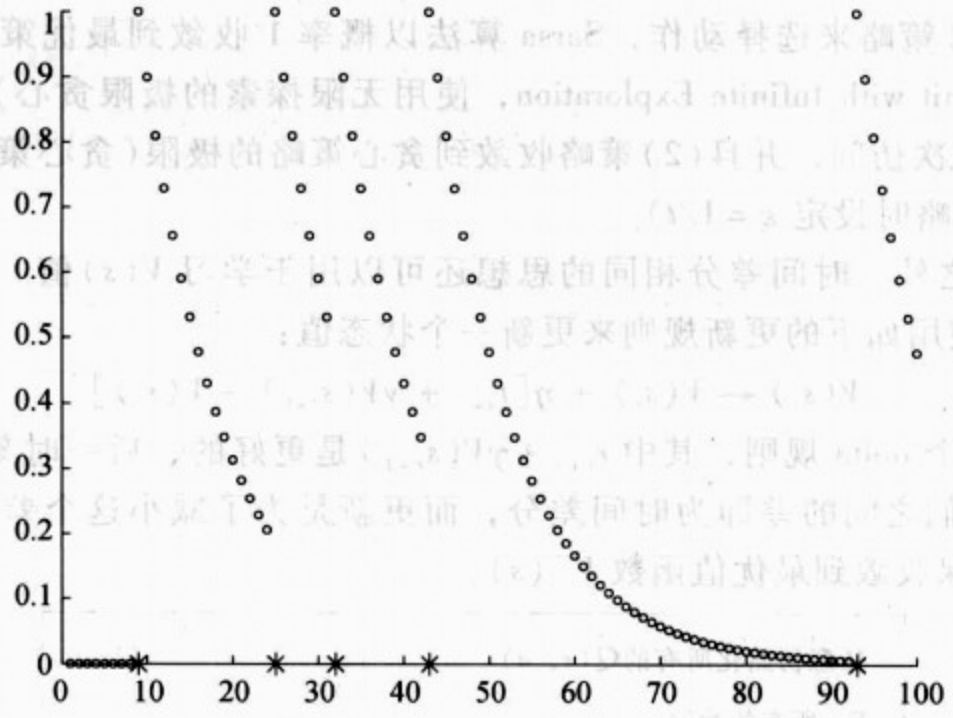


图 16-7 某个值的一个资格迹的例子。访问用星号标记

在 Sarsa 算法中，在时刻 t 的时间误差为：

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \tag{16.18}$$

386 在具有资格迹的 Sarsa 中，称之为 Sarsa(λ)，所有的状态-动作对按下式更新：

$$Q(s,a) \leftarrow Q(s,a) + \eta \delta_t e_t(s,a), \forall s,a \tag{16.19}$$

上式对所有状态-动作对的资格进行更新，更新依赖于其过去出现有多久。λ 值定义了时间信度：如果 $\lambda = 0$，则只进行单步更新。我们在 16.5.3 节讨论的算法就是属于这类，也正因为如此，它们被命名为$Q(0)$、Sarsa(0)或 TD(0)。随着 λ 趋近于 1，之前的更多步骤被考虑在内。当 $\lambda = 1$ 的时候，所有之前的步骤均被考虑在内，并且分配给它们的信度仅以每步 γ 进行下降。在在线更新中，所有的资格值在每步之后立即更新；而在离线更新中，更新累积至片段结束进行单步更新。在线更新花费更多的时间但是收敛的更快。Sarsa(λ)的伪代码图示于 16-8。$Q(\lambda)$和 TD(λ)算法可类似的得到(Sutton 和 Barto1998)。

任意初始化所有的$Q(s, a)$，$e(s, a) \leftarrow 0$，$\forall s, a$
For 所有的片段
初始化 s
使用由Q导出的策略(例如 ε-贪心)选择 a
Repeat
执行动作 a 观测 r 和 s'
使用由Q导出的策略(例如 ε-贪心)选择 a'
$\delta \leftarrow r + \gamma Q(s', a') - Q(s, a)$
$e(s, a) \leftarrow 1$
For 所有的 s，a：
$Q(s, a) \leftarrow Q(s, a) + \eta \delta e(s, a)$
$e(s, a) \leftarrow \gamma \lambda e(s, a)$
$s \leftarrow s'$，$a \leftarrow a'$
Until s 是终止状态

图 16-8 Sarsa(λ)算法

16.6 推广

迄今为止，我们假定$Q(s, a)$值(或者$V(s)$值，如果估计的是状态值)存储在一个查找表中，而我们之前考虑的各种算法称为表格(tabular)算法。这种方法具有几个问题：(1)当状态个数和动作个数很大时，表格的尺寸会变得非常大；(2)状态和动作可能是连续的，例如，将方向盘以某个角度进行调整；而使用表格，将对这些连续值进行离散化，这可能会导致误差；(3)当搜索空间比较大的时候，可能需要非常多的片段才能以可接受的准确程度填满表格的所有项。 387

我们可以将上述问题看作是一个回归问题，来取代使用表格存储Q值。这是一个监督学习问题，其中我们定义一个回归器$Q(s, a \mid \theta)$，将s和a作为输入并通过参数向量θ进行参数化来学习Q值。例如，这个回归器可以是一个人工神经网络，以s和a为输入，一个输出，并以θ为连接权重。

一个好的函数逼近器具有通常意义上的优势并可以解决之前讨论过的如下问题：一个好的逼近可以用一个简单模型来实现，而不必显式存储训练实例；可以使用连续输入；可以推广：如果我们知道相似的(s, a)对之间具有相似的Q值，则我们能够对之前的情况进行推广并产生好的$Q(s, a)$值，即使这一状态之前从未遇到过。

为了可以对回归器进行训练，我们需要一个训练集。在Sarsa(0)的情况下，之前我们看到，我们希望$Q(s_t, a_t)$的值最好接近$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$的值。这样，我们可以形成一个训练集，其中输入是状态-动作对(s_t, a_t)，而要求的输出是$r_{t+1} + \gamma Q(s_{t+1}, a_{t+1})$。我们可以将均方误差写为：

$$E^t(\theta) = [r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]^2 \tag{16.20}$$

可以类似地定义$Q(0)$和TD(0)的训练集。对于后者而言，我们学习$V(s)$，而要求的输出是$r_{t+1} - \gamma V(s_{t+1})$。一旦准备好训练集，我们可以使用任何监督学习算法在训练集上进行学习。

如果我们使用梯度下降方法，像训练神经网络那样，参数向量可更新如下：

$$\Delta\theta = \eta[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \nabla_{\theta_t} Q(s_t, a_t) \tag{16.21}$$

这是单步更新。而在Sarsa(λ)，资格迹也被计算在内：

$$\Delta\theta = \eta\delta_t e_t \tag{16.22}$$ 388

其中时间差分误差是：

$$\delta_t = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$$

并且资格参数向量更新如下：

$$e_t = \gamma\lambda e_{t-1} + \nabla_{\theta_t} Q(s_t, a_t) \tag{16.23}$$

其中e_0为零向量。在表格算法的情况下，会为每对状态-动作存储其资格，因为这些即为(存储为表格)参数。而在使用估计子的情况下，资格是和估计子的参数相关联的。我们也注意到这非常类似于用于稳定反向传播的动量法(11.8.1节)。不同之处在于在动量法中记忆的是先前的权重变化，而这里记忆的是先前的梯度向量。

根据计算$Q(s_t, a_t)$所使用的模型，比如神经网络，我们将其梯度向量插入(16.23)式。

理论上，任何回归方法都可用于训练Q函数，但是针对这一特定任务还是有若干要求：

首先，使用的方法应可以推广，也就是说，我们的确需要保证相似的状态和动作具有相似的Q值。同时像在其他应用中一样，也需要对 s 和 a 有一个好的表示，使得相似性比较明显。其次，增强学习更新以一个接一个的方式提供实例，而不是作为一个整体的训练集，因而学习算法应当有能力进行单个更新来对新的实例进行学习并且不会忘记以前已经学到的东西。例如，只要使用一个很小的学习率，一个后向传播多层感知器可通过一个单独的实例进行训练。或者，可以收集这些实例形成一个训练集来进行学习，但是这种方法减慢了学习速度，因为在一个足够大的样本集被收集到之前不会进行任何学习。

由于这些原因，使用局部学习器对Q值进行学习似乎是一个好主意。在这类方法中，例如径向基函数，信息被局部化并且当对一个新的实例进行学习的时候，学习器的一个局部被更新，而不损坏其他部分的信息。相同的要求也适用于用 $V(s_t \mid \theta)$ 估计状态值。

16.7 部分可观测状态

在某些应用中，智能主体并不确切的知道系统状态。智能主体配备以传感器，传感器返回观测(observation)，而智能主体使用这些观测对系统状态进行估计。比如我们有一个在房间内导航的机器人。这个机器人也许并不知道其在房间内的确切位置，或还有其他什么东西在房间内。机器人可能装备了一个照相机，使用它来记录传感观测。虽然这样并不能告诉机器人其确切的状态但是可以提供关于其可能状态的提示信息。例如，这个机器人可能只知道其右边有一堵墙。

389

这一场景类似于一个马尔可夫决策过程，不同之处是在执行动作 a_t 之后，新的状态 s_{t+1} 是未知的，但是有一个观察 o_{t+1}，它是一个关于 s_t 和 a_t 的随机函数：$p(o_{t+1} \mid s_t, a_t)$。这称为部分可观测马尔可夫决策过程(partially observable MDP，POMDP)。如果 $o_{t+1} = s_{t+1}$，则POMDP简化为MDP。这就像可观测的和隐马尔可夫模型之间的差别，而且它们的求解也类似；也就是说，我们需要从观测来推断状态(或状态的概率分布)并据此执行动作。如果智能主体认为其处于状态 s_1 的概率为0.4而处于状态 s_2 的概率为0.6，则任一动作的值就是0.4乘以在 s_1 状态执行该动作的值加上0.6乘以在 s_2 状态执行该动作的值。

马尔可夫性质对于观测而言并不成立：下一状态的观测并不仅仅依赖于当前的动作和观测。当只存在有限的观测的时候，两个状态表面上看起来可能是一样的，但是实际上却是不同的，而且如果这两个状态要求执行不同的动作，那么就会导致以累积奖励为度量的性能上的损失。智能主体应当以某种方式将过去的轨迹压缩到一个当前的单一状态估计。这些过去的观测还可以通过将观测上的一个过去的窗口作为策略输入而被计算在内，或者使用递归神经网络(11.12.2节)，在不忘记过去的观测的同时维持状态估计。

在任何时候，智能主体都可以对最可能的状态进行计算并执行相应的动作。或者它可以执行动作来收集信息并减小不确定性，例如搜索一个地标，或停下来询问方向等。这意味着信息价值(value of information)(3.6小节)的重要性，并且事实上POMDP可以建模为动态(dynamic)影响图(3.8节)。智能主体根据动作所提供的信息、所产生的奖励大小以及它们如何改变环境状态来选择动作。

为了保持整个过程是马尔可夫的，智能主体维护一个内部的信任状态(belief state) b_t 来对其经历进行总结(见图16-9)。智能主体有一个状态估计子，它基于上一动作 a_t、当前观

测 o_{t+1} 和前一信任状态 b_t 来更新信任状态 b_{t+1}。智能主体还有一个策略 π，与完全可观测环境中所使用的真实状态相反，策略 π 基于这个信任状态来产生下一动作 a_{t+1}。信任状态是给定初始信任状态(在执行任何动作之前)的环境状态和智能主体以往的观测-动作历史(没有遗漏任何可能提高智能主体性能的信息)上的概率分布。在这种情况下，Q学习使用的是信任状态-动作对的值，而非实际的状态-动作对的值： 390

$$Q(b_t,a_t) = E[r_{t+1}] + \gamma \sum_{b_{t+1}} P(b_{t+1} \mid b_t,a_t) V(b_{t+1}) \qquad (16.24)$$

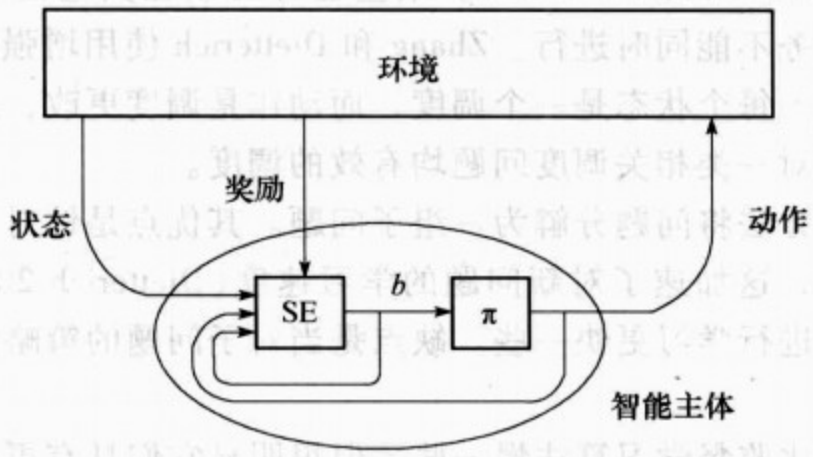


图 16-9 在部分可观测环境中，智能主体具有一个状态估计子(SE)对内部信任状态 b 进行维护并且策略 π 根据这些信任状态产生动作

Kaelbling、Littman 和 Cassandra 1998 给出了一个算法，但是很不幸，由于其复杂度很高，该算法只能对仅有几十个状态的系统精确求解。否则，必须借助于价值函数 $V(b_t+1)$ 的近似求解算法；Hauskrecht 2000 给出了此类算法的综述。

16.8 注释

关于增强学习的更多信息可以在 Sutton 和 Barto(1998)的教科书中找到，该书讨论了增强学习的各个方面、学习算法以及若干应用。而 Kaelbling，Littman 和 Moore 1996 是增强学习的全面介绍。 391

Bertsekas 1987 以及 Bertsekas 和 Tsitsiklis 1996 讨论了动态规划方法，而Q学习可以看作是动态规划的随机近似(Jaakkola、Jordan 和 Singh 1994)。增强学习相对于经典动态规划具有两个优点：首先，在学习期间，增强学习可专注于空间的重要部分而忽略其他部分；其次，增强学习可以使用函数逼近方法来表示知识，进而得以推广和更快地学习。

一个相关的领域是学习自动机(learning automata)(Narendra 和 Thathachar 1974)它是一个有限状态机器，通过“试错”解决类似于 K-臂赌博机问题。我们这里所讨论的场景同样也是最优控制的课题，其中一个控制器(智能主体)在设施(环境)中执行动作来最小化系统开销(最大化奖励)。

最早使用时间差分方法的是 Samuel 写于 1959 年的跳棋游戏程序(Sutton 和 Barto 1998)。对于一个游戏中每对相继的位置，通过棋盘评估函数对两个棋盘状态进行评估，进而引发一个更新来减小它们之间的差异。关于游戏方面的研究工作很多，因为其兼具易于定义和挑战性的特点。对一个类似象棋的游戏的模拟也易于进行：允许的棋步可以形式化而且目标状态清晰。尽管定义这样一个游戏很简单，但是以专家级别进行游戏却非常困难。

增强学习最令人印象深刻的应用是TD-Gammon程序。该程序通过和自身进行对弈来学习下西洋双陆棋(Tesauro 1995)。它优于同样由Tesauro开发的neruogammon程序，后者基于与专家对弈，以监督学习方式进行训练。西洋双陆棋是大约有10^{20}种状态的复杂任务，并存在由于掷骰子而产生的随机性。使用TD(λ)算法，TD-Gammon程序在经过和自身副本1 500 000次对弈后达到了大师级水平。

另一个有趣的应用是作业车间调度(job shop scheduling)问题或寻找满足时间和资源约束的任务调度问题(Zhang和Dietterich 1996)。某些任务必须在其他任务开始之前完成，并且需要相同资源的两个任务不能同时进行。Zhang和Dietterich使用增强学习很快找到了满足约束并且较短的调度方式。每个状态是一个调度，而动作是调度更改，最终程序找到的不仅是一个好的调度，而且是对一类相关调度问题均有效的调度。

392 最近提出了层次化方法将问题分解为一组子问题。其优点是针对子问题学习而得到的策略可在多个问题上共享，这加速了对新问题的学习速度(Dietterich 2000)。每个子问题都更简单，并且对它们单独进行学习更快一些。缺点是当对子问题的策略进行组合时，所得的策略可能是次最优的。

尽管增强学习算法比监督学习算法慢一些，但很明显它们具有更广泛的应用并具有构建更好学习机器的潜力(Ballard1997)。它们不需要任何监督，因而可能实际上更好一些，因为不会被老师误导。例如，Tesauro的TD-Gammon程序在某些情况下所走的棋步比最好的棋手所走的棋步还要好。增强学习领域发展迅速，因而我们可以期待在不远的将来看到其他引人注目的成果。

16.9 习题

1. 给定图16-10的网格世界，如果到达目标的奖励为100并$\gamma = 0.9$，手工计算$Q^*(s, a)$，$V^*(S)$以及最优策略的动作。

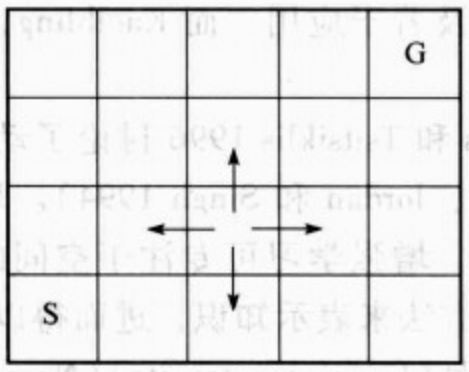


图16-10 网格世界。智能主体始于S，可以向四个罗盘方向移动。目标状态为G

2. 以练习1中相同的配置，使用Q学习算法学习最优策略。
3. 在练习1中，如果在右下角加入另一个目标状态，最优策略将如何改变？如果在右下角
393 的状态定义奖励为-100(非常坏的状态)将发生什么？
4. 作为对$\gamma < 1$的替代，有$\gamma = 1$并且所有中间状态(非目标)具有一个负的奖励$-c$。这二者有何差异？
5. 在练习1中，假设到达目标的奖励服从均值100和方差40的正态分布。同时假设动作也是随机的，即当机器人向一个方向前进的时候，它以0.5的概率向预定的方向前进同时

以 0.25 的概率向两个横向方向之一前进。在这种情况下，学习$Q(s, a)$。

6. 假设我们想要使用 TD(λ)算法对状态值函数 $V(s)$进行估计。推导出其表值迭代更新。
7. 使用(16.22)式，推导出使用多层感知器估计Q的权重更新公式。
8. 给出一个可用 POMDP 建模的增强学习应用的例子。定义其中的状态、动作、观测和奖励。

16.10 参考文献

Ballard, D. H. 1997. *An Introduction to Natural Computation.* Cambridge, MA: The MIT Press.

Bellman, R. E. 1957. *Dynamic Programming.* Princeton: Princeton University Press.

Bertsekas, D. P. 1987. *Dynamic Programming: Deterministic and Stochastic Models.* New York: Prentice Hall.

Bertsekas, D. P., and J. N. Tsitsiklis. 1996. *Neuro-Dynamic Programming.* Belmont, MA: Athena Scientific.

Dietterich, T. G. 2000. "Hierarchical Reinforcement Learning with the MAXQ Value Decomposition." *Journal of Artificial Intelligence Research* 13: 227 – 303.

Hauskrecht, M. 2000. "Value- Function Approximations for Partially Observable Markov Decision Processes." *Journal of Artificial Intelligence Research* 13: 33 – 94.

Jaakkola, T., M. I. Jordan, and S. P. Singh. 1994. "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms." *Neural Computation* 6: 1185 – 1201.

Kaelbling, L. P., M. L. Littman, and A. R. Cassandra. 1998. "Planning and Acting in Partially Observable Stochastic Domains." *Artificial Intelligence* 101: 99 – 134.

Kaelbling, L. P., M. L. Littman, and *A.* W. Moore. 1996. "Reinforcement Learning: A Survey." *Journal of Artificial Intelligence Research* 4: 237 – 285.

Narendra, K. S., and M. A. L. Thathachar. 1974. "Learning Automata—A Survey." *IEEE Transactions on Systems, Man, and Cybernetics* 4: 323 – 334.

Sutton, R. S. 1988. "Learning to Predict by the Method of Temporal Differences." *Machine Learning* 3: 9 – 44.

Sutton, R. S., and A. G. Barto. 1998. *Reinforcement Learning: An Introduction.* Cambridge, MA: The MIT Press.

Tesauro, G. 1995. "Temporal Difference Learning and TD-Gammon." *Communications of the ACM* 38(3): 58 – 68.

Watkins, C. J. C. H., and P. Dayan. 1992. "Q-learning." *Machine Learning* 8: 279 – 292.

Zhang, W., and T. G. Dietterich. 1996. "High-Performance Job-Shop Scheduling with a Time-Delay TD(λ) Network." In *Advances in Neural Information Processing Systems* 8, ed. D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, 1024 – 1030. Cambridge, MA: The MIT Press.

附录A 概 率 论

我们简略回顾概率论原理、随机变量概念和实例分布。

A.1 概率论原理

随机试验是其结果不能提前以确定的方式预测的试验(Ross 1987; Casella 和 Berger 1990)。所有可能的结果的集合称作样本空间 S。一个样本空间是离散的，如果它由结果的有限(或可数无限)集组成；否则是连续的。S 的任意子集是一个事件。事件是集合，并且我们可以谈论它们的补、交、并等。

概率的一种解释是频率：当一个试验在完全相同的条件下不断重复时，对于任意事件 E，结果在 E 中的次数所占的比例趋向于某个常数值。这个常数极限频率是事件的概率，而我们把它记作 $P(E)$。

有时，概率可解释成可信程度。例如，当我们说土耳其赢得 2006 年足球世界杯冠军的概率时，我们并不是指出现的频率，因为 2006 年足球世界杯只进行一次，并且(在写本书时)它还未进行。在这种情况下，我们的意思是我们主观相信该事件出现的程度。由于是主观的，因此对同一事件，不同的人可能指派不同的概率。

A.1.1 概率论公理

公理确保随机试验中指派的概率可以解释成相对频率，并且这些指派符合我们对相对频率之间关系的直观理解：

1. $0 \leqslant P(E) \leqslant 1$。如果 E_1 是不可能出现的事件，则 $P(E_1)=0$。如果 E_2 是一定出现的事件，则 $P(E_2)=1$。

2. 如果 S 是包含所有可能结果的样本空间，则 $P(S)=1$。

3. 如果 E_i，$i=1$，…，n，是互斥的(即如果它们不可能同时出现：$E_i \cap E_j = \varnothing$，$i \neq j$，其中∅是不包含任何可能结果的空事件)，则我们有

$$P\left(\bigcup_{i=1}^{n} E_i\right) = \sum_{i=1}^{n} P(E_i) \tag{A.1}$$

例如，设 E^c 表示 E 的补，由不在 E 中的 S 中所有可能的结果组成，我们有 $E \cap E^c = \varnothing$，并且

$$P(E \cup E^c) = P(E) + P(E^c) = 1$$
$$P(E^c) = 1 - P(E)$$

如果 E 和 F 的交非空，则我们有

$$P(E \cup F) = P(E) + P(F) - P(E \cap F) \tag{A.2}$$

A.1.2 条件概率

$P(E \mid F)$是给定事件 F 出现，事件 E 出现的概率，并由下式给出

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)} \tag{A.3}$$

知道事件 F 出现将样本空间缩小到 F，而 E 也出现的部分为 $E \cap F$。注意，(A.3)式仅当 $P(F)>0$ 时才有定义。由于∩是可交换的，我们有

$$P(E \cap F) = P(E \mid F)P(F) = P(F \mid E)P(E)$$

398

由此得到贝叶斯公式(Bayes' formula)：

$$P(F \mid E) = \frac{P(E \mid F)P(F)}{P(E)} \tag{A.4}$$

当 F_i 互斥并穷举时，即当 $\bigcup_{i=1}^{n} F_i = S$ 时

$$E = \bigcup_{i=1}^{n} E \cap F_i$$

$$P(E) = \sum_{i=1}^{n} P(E \cap F_i) = \sum_{i=1}^{n} P(E \mid F_i)P(F_i) \tag{A.5}$$

贝叶斯公式使得我们可以有

$$P(F_i \mid E) = \frac{P(E \cap F_i)}{P(E)} = \frac{P(E \mid F_i)P(F_i)}{\Sigma_j P(E \mid F_j)P(F_j)} \tag{A.6}$$

如果 E 和 F 是独立的(independent)，我们有 $P(E \mid F) = P(E)$，因此

$$P(E \cap F) = P(E)P(F) \tag{A.7}$$

也就是说，F 是否出现的知识并不改变 E 出现的概率。

A.2 随机变量

随机变量(random variable)是一个函数，它对随机试验的样本空间中的每个结果指派一个数。

A.2.1 概率分布与密度函数

对于任意实数值 a，随机变量 X 的概率分布函数 $F(\cdot)$是

$$F(a) = P\{X \leqslant a\} \tag{A.8}$$

并且我们有

$$P\{a < X \leqslant b\} = F(b) - F(a) \tag{A.9}$$

如果 X 是离散的随机变量，则

$$F(a) = \sum_{\forall x \leqslant a} P(x) \tag{A.10}$$

399

其中 $P(\cdot)$是概率质量函数(probability mass function)，定义为 $P(a) = P\{X = a\}$。如果 X 是连续的随机变量，则 $p(\cdot)$是概率密度函数(probability density function)，使得

$$F(a) = \int_{-\infty}^{a} p(x)dx \tag{A.11}$$

A.2.2 联合分布与密度函数

在特定的试验中，我们可能对两个或多个随机变量之间的关系感兴趣，并且我们使用 X 和 Y 的联合(joint)概率分布和密度函数，满足

$$F(x,y) = P\{X \leqslant x, Y \leqslant y\} \tag{A.12}$$

单个边缘(marginal)分布和密度可以通过边缘化来计算，即在自由变量上求和：

$$F_X(x) = P\{X \leqslant x\} = P\{X \leqslant x, Y \leqslant \infty\} = F(x,\infty) \tag{A.13}$$

在离散情况下，我们有

$$P(X = x) = \sum_j P(x,y_j) \tag{A.14}$$

而在连续情况下，我们有

$$px(x) = \int_{-\infty}^{\infty} p(x,y)dy \tag{A.15}$$

如果 X 和 Y 是独立的(independent)，我们有

$$p(x,y) = p_X(x)p_Y(y) \tag{A.16}$$

这些都能够以直截了当的方式推广到多于两个随机变量的情况。

A.2.3 条件分布

当 X 和 Y 是随机变量时，

400

$$P_{X|Y}(x|y) = P\{X = x | Y = y\} = \frac{P\{X = x | Y = y\}}{P\{Y = y\}} = \frac{P(x,y)}{P_Y(y)} \tag{A.17}$$

A.2.4 贝叶斯规则

当两个随机变量联合分布，其中一个的值已知时，另一个取给定值的概率可以使用贝叶斯规则计算：

$$P(y|x) = \frac{P(x|y)P_Y(Y)}{P_X(x)} = \frac{P(x|y)P_Y(y)}{\sum_y P(x|y)P_Y(y)} \tag{A.18}$$

换言之，

$$\text{后验} = \frac{\text{似然} \times \text{先验}}{\text{证据}} \tag{A.19}$$

注意，分母通过在所有可能的 y 值上对分子求和(或积分，如果 y 是连续的)得到。$p(y|x)$ 的“形状”取决于分子，分母作为规范化因子确保 $p(y|x)$ 的和为 1。通过考虑 x 提供的信息，贝叶斯规则使得我们将一个先验概率修改为后验概率。

贝叶斯规则反转依赖性，如果 $p(x|y)$ 已知，使得我们可以计算 $p(y|x)$。假设 y 是 x 的“原因”，如 y 是度暑假，x 是被晒黑。则 $p(x|y)$ 是已知某人度暑假，他被晒黑的概率。这是

因果(causal)(或预测)方法。贝叶斯规则允许我们使用诊断(diagnostic)方法来计算 $p(y \mid x)$：即某人知道会被晒黑而去度暑假的概率。$p(y)$是任何人去度暑假的概率，而 $p(x)$是任何人被晒黑的概率，包括度暑假和不度暑假的人。

A.2.5 期望

随机变量 X 的期望(expectation)、期望值(expected value)或均值(mean)记作 $E[X]$，是大量试验中 X 的平均值：

$$E[X] = \begin{cases} \sum_i x_i P(x_i) & \text{如果 } X \text{ 是离散的} \\ \int x p(x)\,dx & \text{如果 } X \text{ 是连续的} \end{cases} \tag{A.20}$$

它是加权平均，其中每个值被 X 取该值的概率加权。它具有如下性质(a，$b \in \Re$)：

$$E[aX + b] = aE[X] + b$$
$$E[X + Y] = E[X] + E[Y] \tag{A.21}$$

401

对于任意实数值函数 $g(\cdot)$，期望值是

$$E[g(X)] = \begin{cases} \sum_i g(x_i) P(x_i) & \text{如果 } X \text{ 是离散的} \\ \int g(x) p(x)\,dx & \text{如果 } X \text{ 是连续的} \end{cases} \tag{A.22}$$

一种特例 $g(x) = x^n$，称作 X 的 n 阶矩，定义为

$$E[X^n] = \begin{cases} \sum_i x_i^n P(x_i) & \text{如果 } X \text{ 是离散的} \\ \int x^n p(x)\,dx & \text{如果 } X \text{ 是连续的} \end{cases} \tag{A.23}$$

均值(mean)是一阶矩并记作 μ。

A.2.6 方差

方差(variance)度量 X 在期望值附近的变化。如果 $\mu \equiv E[X]$，则方差定义为

$$\mathrm{Var}(X) = E[(X - \mu)^2] = E[X^2] - \mu^2 \tag{A.24}$$

方差是二阶矩减去一阶矩的平方。方差记作 σ^2，具有如下性质(a，$b \in \Re$)：

$$\mathrm{Var}(aX + b) = a^2 \mathrm{Var}(X) \tag{A.25}$$

$\sqrt{\mathrm{Var}(X)}$称作标准差(standard deviation)，记作 σ。标准差具有和 X 相同的单位，并且比方差容易解释。

协方差(covariance)指示两个随机变量之间的关系。如果 X 的出现使得 Y 更可能出现，则协方差为正；如果 X 的出现使得 Y 更不可能发生，则协方差为负；如果没有依赖性，则协方差为 0。

$$\mathrm{Cov}(X,Y) = E[(X - \mu_X)(Y - \mu_Y)] = E[XY] - \mu_X \mu_Y \tag{A.26}$$

其中 $\mu_X \equiv E[X]$，$\mu_Y \equiv E[Y]$。一些其他性质是

$$\mathrm{Cov}(X,Y) = \mathrm{Cov}(Y,X)$$
$$\mathrm{Cov}(X,X) = \mathrm{Var}(X)$$

$$\mathrm{Cov}(X+Z,Y) = \mathrm{Cov}(X,Y) + \mathrm{Cov}(Z,Y)$$

$$\mathrm{Cov}\left(\sum_i X_i, Y\right) = \sum_i \mathrm{Cov}(X_i,Y) \tag{A.27}$$

$$\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\mathrm{Cov}(X,Y) \tag{A.28}$$

402

$$\mathrm{Var}\left(\sum_i X_i\right) = \sum_i \mathrm{Var}(X_i) + \sum_i \sum_{j\neq i} \mathrm{Cov}(X_i,Y_j) \tag{A.29}$$

如果 X 和 Y 是独立的，则 $E[XY]=E[X]E[Y]=\mu_X\mu_Y$，并且 $\mathrm{Cov}(X,Y)=0$。这样，如果 X_i 是独立的，则

$$\mathrm{Var}\left(\sum_i X_i\right) = \sum_i \mathrm{Var}(X_i) \tag{A.30}$$

相关性(correlation)是一个规范化的、维无关的量，其值总是在 -1 和 1 之间：

$$\mathrm{Corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}} \tag{A.31}$$

A.2.7 弱大数定律

设 $\mathcal{X}=\{X^t\}_{t=1}^N$ 是独立的同分布的(iid)随机变量的集合，每个都具有均值 μ 和有限方差 σ^2。则对于任意 $\varepsilon>0$，

$$P\left\{\left|\frac{\sum_t X^t}{N} - \mu\right| > \varepsilon\right\} \to 0 \quad \text{随 } N\to\infty \tag{A.32}$$

也就是说，随着 N 趋向于无穷大，N 个试验的平均值趋向于均值。

A.3 特殊的随机变量

有一些类型的随机变量频繁出现，因此对它们命名。

A.3.1 伯努利分布

试验进行，其结果或者“成功”，或者“失败”。随机变量 X 是一个 0/1 指示变量，并且对于成功结果取值 1，否则为 0。p 是试验结果为成功的概率。则

$$P\{X=1\} = p, \text{而 } P\{X=0\} = 1-p \tag{A.33}$$

这等价于

$$P\{X=i\} = p^i(1-p)^{1-i}, i=0,1 \tag{A.34}$$

如果 X 是伯努利变量，则它的期望值和方差是

403

$$E[X] = p, \mathrm{Var}(X) = p(1-p) \tag{A.35}$$

A.3.2 二项分布

如果做了 N 次相同的、独立的伯努利试验，代表 N 次试验中成功次数的随机变量 X 是二项分布的。i 次成功的概率为

$$P\{X=i\} = C_N^i p^i (1-p)^{N-i}, i=0\cdots N \tag{A.36}$$

如果 X 是二项的，则它的期望值和方差为

$$E[X] = Np, \quad \mathrm{Var}(X) = Np(1-p) \tag{A.37}$$

A.3.3 多项分布

考虑伯努利分布的推广。其中，取代两种状态，随机事件的结果是 K 个互斥、穷举状态之一，每个具有出现概率 p_i，其中 $\sum_{i=1}^{K} p_i = 1$。假设做了 N 次这样的试验，其中结果 i 出现 N_i 次，满足 $\sum_{i=1}^{N} N_i = N$。则 N_1，N_2，…，N_K 的联合分布是多项分布：

$$P(N_1, N_2, \cdots, N_K) = N! \prod_{i=1}^{K} \frac{p_i^{N_i}}{N_i!} \tag{A.38}$$

当 $N=1$ 时是一种特殊情况：只做了一次试验。于是 N_i 是0/1指示变量，其中只有一个为1，其余均为0。(A.38)式归约为

$$P(N_1, N_2, \cdots, N_K) = \prod_{i=1}^{K} p_i^{N_i} \tag{A.39}$$

A.3.4 均匀分布

X 均匀地分布在区间[a，b]上，如果它的密度函数由下式给定

$$p(x) = \begin{cases} \dfrac{1}{b-a} & \text{如果 } a \leqslant x \leqslant b \\ 0 & \text{否则} \end{cases} \tag{A.40}$$

如果 X 是均匀的，则它的期望值和方差为

$$E[X] = \frac{a+b}{2}, \mathrm{Var}(X) = \frac{(b-a)^2}{12} \tag{A.41}$$

404

A.3.5 正态(高斯)分布

X 是均值为 μ、方差为 σ^2 的正态或高斯分布，记作$\mathcal{N}(\mu, \sigma^2)$，如果它的密度函数是

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \quad -\infty < x < \infty \tag{A.42}$$

许多随机现象都遵守钟形正态分布，或至少近似地遵守正态分布；许多自然观测都可以看作连续的、典型值的稍微不同的版本——这或许是将它称作正态(normal)分布的原因。在这种情况下，μ 定义典型值，而 σ 定义典型值附近实例变化的大小。

68.27%的值落在($\mu-\sigma$，$\mu+\sigma$)中，95.45%的值落在($\mu-2\sigma$，$\mu+2\sigma$)中，99.73%的值落在($\mu-3\sigma$，$\mu+3\sigma$)中。这样，$P\{|x-\mu|<3\sigma\} \approx 0.99$。实践中，如果 $x<\mu-3\sigma$ 或 $x>\mu+3\sigma$，则 $p(x)\approx 0$。Z是单位正态分布，即$\mathcal{N}(0, 1)$(见图 A-1)，并且它的密度记作

$$p_Z(x) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{x^2}{2}\right] \tag{A.43}$$

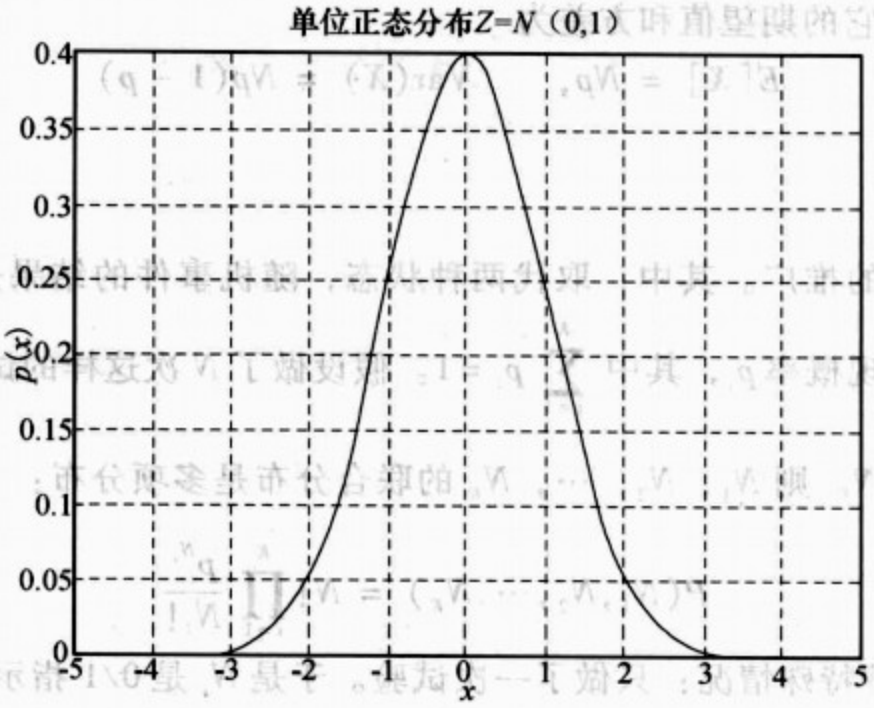


图 A-1 单位正态分布Z的概率密度函数

如果 $X \sim \mathcal{N}(\mu, \sigma^2)$并且 $Y = aX + b$，则 $Y \sim \mathcal{N}(a\mu + b, a^2\sigma^2)$。独立的正态变量的和也是正态的，其中 $\mu = \sum_i \mu_i$，$\sigma^2 = \sum_i \sigma_i^2$。如果 X 是$\mathcal{N}(\mu, \sigma^2)$，则

$$\frac{X - \mu}{\sigma} \sim Z \tag{A.44}$$

这称作 z-标准化。

设 X_1，X_2，…，X_N 是 iid 随机变量，都具有均值 μ 和方差 σ^2。则中心极限定理表明对于大的 N，分布

$$X_1 + X_2 + \cdots + X_N \tag{A.45}$$

近似于$\mathcal{N}(N\mu, N\sigma^2)$。例如，如果 X 是参数为(N, p)的二项分布，则 X 可以写成 N 个伯努利试验的和，并且$(X - Np)/\sqrt{Np(1-p)}$是近似单位正态的。

中心极限定理也用来在计算机上产生正态分布的随机变量。程序设计语言具有一些子程序，返回[0，1]上均匀分布的(伪)随机数。当 U_i 是这样的随机变量时，$\sum_{i=1}^{12} U_i - 6$ 近似于Z。

设 $X' \sim \mathcal{N}(\mu, \sigma^2)$。估计样本均值

$$m = \frac{\sum_{i=1}^{N} X'}{N} \tag{A.46}$$

也是正态的，均值为 μ，而方差为 σ^2/N。

A.3.6 卡方分布

如果 Z_i 是独立的单位正态随机变量，则

$$X = Z_1^2 + Z_2^2 + \cdots + Z_n^2 \tag{A.47}$$

是自由度为 n 的卡方分布，即 $X \sim \chi_n^2$，其中

$$E[X] = n, \mathrm{Var}(X) = 2n \tag{A.48}$$

当 $X^t \sim \mathcal{N}(\mu, \sigma^2)$时，估计样本方差为

$$S^2 = \frac{\sum_t (X^t - m)^2}{N-1} \tag{A.49}$$

并且我们有

$$(N-1)\frac{S^2}{\sigma^2} \sim \chi^2_{N-1} \tag{A.50}$$

还知道 m 和 S^2 是独立的。

405
~
406

A.3.7 t 分布

如果 $Z \sim \mathcal{Z}$和 $X \sim \chi^2_n$ 是独立的，则

$$T_n = \frac{Z}{\sqrt{X/n}} \tag{A.51}$$

是自由度为 n 的 t-分布，其中

$$E[T_n] = 0, n > 1, \mathrm{Var}(T_n) = \frac{n}{n-2}, n > 2 \tag{A.52}$$

像单位正态密度一样，t-分布在 0 周围是对称的。随着 n 越来越大，t 密度变得越来越像正态分布，区别是 t-分布具有较粗的尾部，表明比正态分布具有更大的可变性。

A.3.8 F 分布

如果 $X_1 \sim \chi^2_n$ 和 $X_2 \sim \chi^2_m$ 分别是自由度为 n 和 m 的卡方随机变量，则

$$F_{n,m} = \frac{X_1/n}{X_2/m} \tag{A.53}$$

是自由度为 n 和 m 的 F - 分布，其中

$$E[F_{n,m}] = \frac{m}{m-2}, m > 2, \mathrm{Var}(F_{n,m}) = \frac{m^2(2m+2n-4)}{n(m-2)^2(m-4)}, m > 4 \tag{A.54}$$

A.4 参考文献

Casella, G., and R. L. Berger. 1990. *Statistical Inference*. Belmont, CA: Duxburry.

Ross, S. M. 1987. *Introduction to Probability and Statistics for Engineers and Scientists*. New York: Wiley.

407

索　引

索引中标注的页码为英文原书页码，与书中边栏的页码一致。

D

E

F

G

H

I

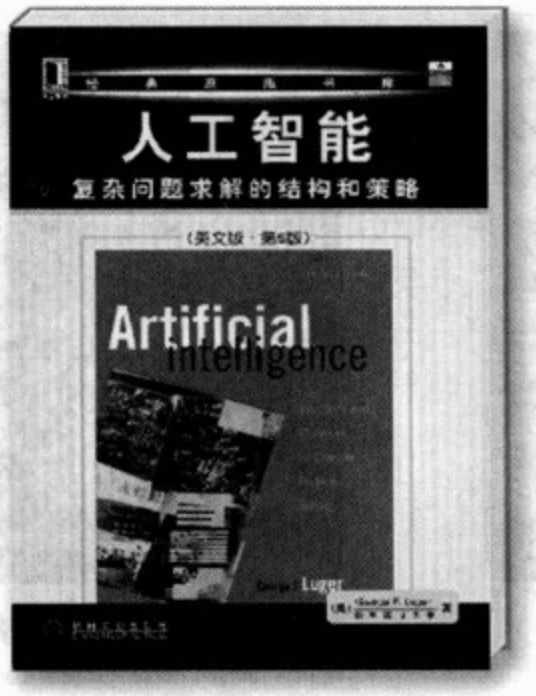
J

K

S

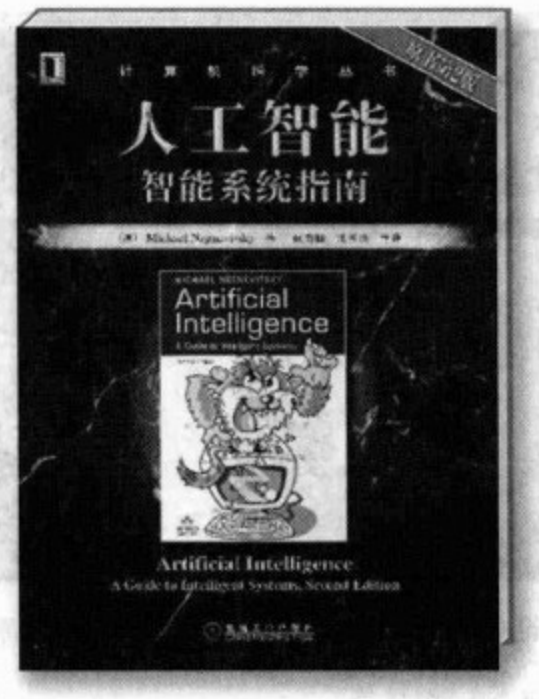
T

延伸阅读



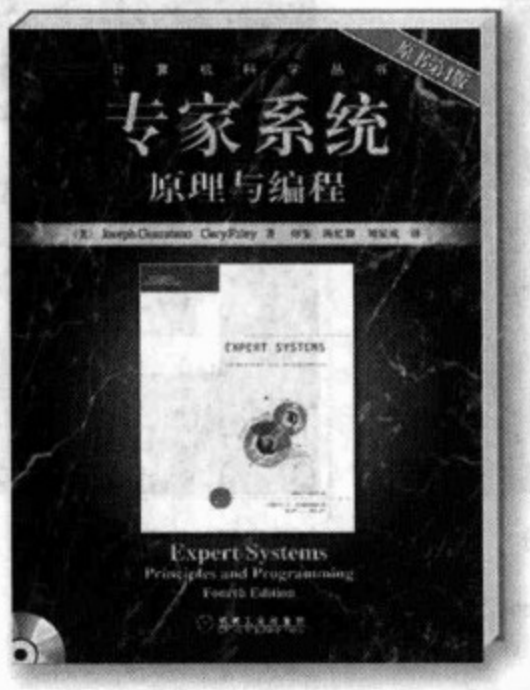
人工智能复杂问题求解的结构和策略
(英文版·第6版)

作 者：George F. Luger
书 号：978-7-111-25656-4
定 价：46.00元



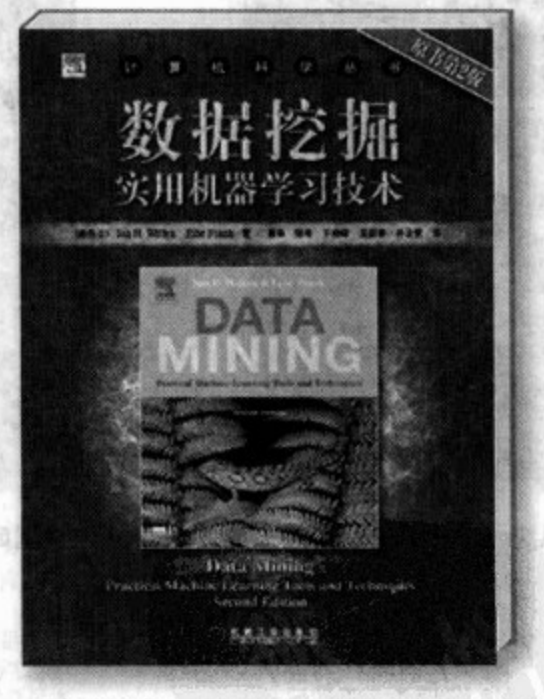
人工智能 智能系统指南（原书第2版）

作 者：Michael Negnevitsky
译 者：顾力栩 沈晋惠 等
书 号：978-7-111-20212-7
定 价：36.00元



专家系统：原理与编程(原书第4版)

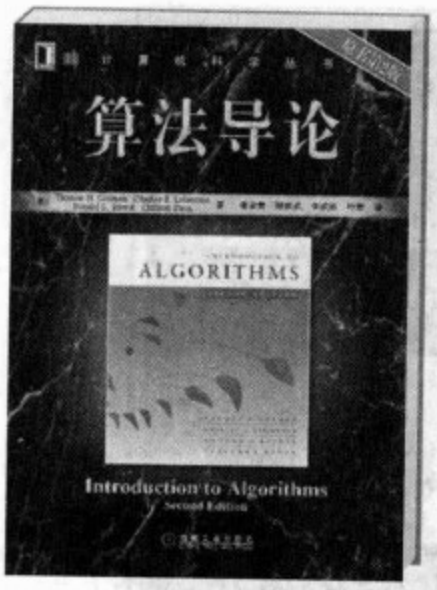
作 者：Joseph Giarratano；Gary D.Riley
译 者：印鉴 陈忆群 刘星成
书 号：978-7-111-19203-6
定 价：65.00元



数据挖掘：实用机器学习技术（原书第2版）

作 者：Ian H.Witten, Eibe Frank
译 者：董琳 邱泉 于晓峰 吴韶群 孙立骏
书 号：978-7-111-18205-7
定 价：48.00元

经典推荐

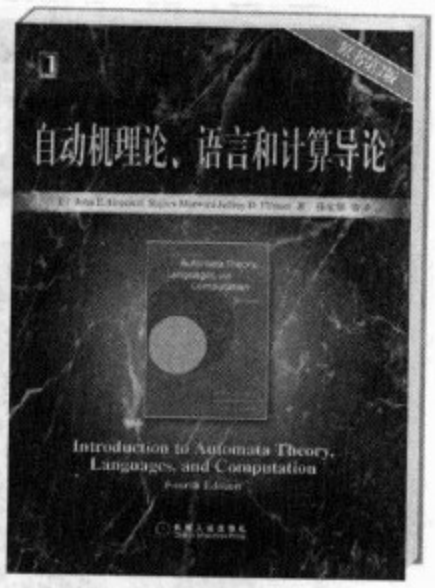


算法导论 (原书第2版)

作　者：Thomas H.Cormen 等
译　者：潘金贵 顾铁成 等
书　号：7-111-18777-6
定　价：85.00元
■2006、2007 CSDN《程序员》杂志评选的十大IT好书之一，算法中的经典权威之作

编译原理 (第2版)

作　者：Alfred V.Aho,Monica S.Lam, Ravi Sethi,Jeffrey D.Ullman
译　者：赵建华 等
书　号：978-7-111-25121-7
■编译领域无可替代的经典著作，被广大计算机专业人士誉为"龙书"



自动机理论、语言和计算导论 (原书第3版)

作　者：John E.Hopcroft,Rajeev Motwani, Jeffrey D.Ullman
译　者：孙家骕 等
中文版：978-7-111-24035-8，49.00元
英文版：978-7-111-22392-4，59.00元
■1996年图灵奖得主经典巨著升级版

分布式系统：概念与设计(原书第4版)

作　者：George Coulouris, Jean Dollimore, Tim Kindberg
译　者：金蓓弘 曹冬磊
中文版：978-7-111-22438-9，69.00元
英文版：7-111-17366-X，89.00元
■本书是衡量所有其他分布式系统教材的标准

数据库系统概念(原书第5版)

作　者：Abraham Silberschatz, Henry F.Korth, S. Sudarshan
译　者：杨冬青 马秀莉 唐世渭
中文版：7-111-19687-2，69.50元
本科教学版：978-7-111-23422-7，45.00元
■数据库系统方面的经典教材，被美誉为"帆船书"

软件工程：实践者的研究方法(原书第6版)

作　者：Roger S.Pressman
译　者：郑人杰 等
中文版：7-111-19400-4，69.00元
本科教学版：978-7-111-23443-2，49.00元
英文精编版：978-7-111-24138-6，65.00元
■全球上百所大学和学院采用，最受欢迎的软件工程指南

教师服务登记表

尊敬的老师：

您好！感谢您购买我们出版的 _____ 教材。

机械工业出版社华章公司本着为服务高等教育的出版原则，为进一步加强与高校教师的联系与沟通，更好地为高校教师服务，特制此表，请您填妥后发回给我们，我们将定期向您寄送华章公司最新的图书出版信息。为您的教材、论著或译著的出版提供可能的帮助。欢迎您对我们的教材和服务提出宝贵的意见，感谢您的大力支持与帮助！

个人资料（请用正楷完整填写）

<table>
<tr><td>教师姓名</td><td></td><td>□先生
□女士</td><td>出生年月</td><td></td><td>职务</td><td></td><td colspan="2">职称：□教授 □副教授
□讲师 □助教 □其他</td></tr>
<tr><td>学校</td><td colspan="3"></td><td>学院</td><td colspan="2"></td><td>系别</td><td></td></tr>
<tr><td rowspan="2">联系
电话</td><td colspan="3" rowspan="2">办公：
宅电：
移动：</td><td>联系地址
及邮编</td><td colspan="3"></td></tr>
<tr><td>E-mail</td><td colspan="3"></td></tr>
<tr><td>学历</td><td></td><td>毕业院校</td><td colspan="2"></td><td>国外进修及讲学经历</td><td colspan="3"></td></tr>
<tr><td>研究领域</td><td colspan="8"></td></tr>
<tr><td colspan="2">主讲课程</td><td colspan="3">现用教材名</td><td>作者及
出版社</td><td>共同授
课教师</td><td colspan="2">教材满意度</td></tr>
<tr><td colspan="2">课程：
□专 □本 □研 □MBA
人数： 学期：□春□秋</td><td colspan="3"></td><td></td><td></td><td colspan="2">□满意 □一般
□不满意 □希望更换</td></tr>
<tr><td colspan="2">课程：
□专 □本 □研 □MBA
人数： 学期：□春□秋</td><td colspan="3"></td><td></td><td></td><td colspan="2">□满意 □一般
□不满意 □希望更换</td></tr>
<tr><td colspan="9">样书申请</td></tr>
<tr><td colspan="3">已出版著作</td><td colspan="3"></td><td>已出版译作</td><td colspan="2"></td></tr>
<tr><td colspan="5">是否愿意从事翻译/著作工作 □是 □否</td><td>方向</td><td colspan="3"></td></tr>
<tr><td>意见和建议</td><td colspan="8"></td></tr>
</table>

填妥后请选择以下任何一种方式将此表返回：（如方便请赐名片）

地 址：北京市西城区百万庄南街1号 华章公司营销中心 邮编：100037

电 话：(010) 68353079 88378995 传真：(010)68995260

E-mail:hzedu@hzbook.com markerting@hzbook.com 图书详情可登录http://www.hzbook.com网站查询

机器学习导论

机器学习的目标是对计算机编程，以便使用样本数据或以往的经验来解决给定的问题。已经有许多机器学习的成功应用，包括分析以往销售数据来预测客户行为，人脸识别或语音识别，优化机器人行为以便使用最少的资源来完成任务，以及从生物信息数据中提取知识的各种系统。为了对机器学习问题和解进行统一的论述，本书讨论了机器学习在统计学、模式识别、神经网络、人工智能、信号处理、控制和数据挖掘等不同领域的应用。对所有学习算法都进行了解释，以便读者可以容易地将书中的公式转变为计算机程序。本书可作为高等院校计算机相关专业高年级本科生和研究生的教材，也可供研究机器学习方法的技术人员参考。

本书对机器学习的定义和应用实例进行了介绍，涵盖了监督学习、贝叶斯决策理论、参数方法、多元方法、维度归约、聚类、非参数方法、决策树、线性判别式、多层感知器、局部模型、隐马尔可夫模型、分类算法评估和比较、组合多学习器以及增强学习等。

作者简介

Ethem Alpaydın 是土耳其伊斯坦布尔博阿齐奇大学计算机工程系的教授。于1990年在洛桑联邦理工学院获博士学位，之后先后在美国麻省理工和伯克利大学工作和进行博士后研究。Ethem博士主要从事机器学习方面的研究，是剑桥大学的《The Computer Journal》杂志编委和Elsevier的《Pattern Recognition》杂志的副主编。2001年和2002年，Ethem博士先后获得土耳其科学院青年科学家奖和土耳其科学与技术研究委员会科学奖。

投稿热线：(010) 88379604
购书热线：(010) 68995259, 68995264
读者信箱：hzjsj@hzbook.com

华章网站 http://www.hzbook.com

网上购书：www.china-pub.com

封面设计：锡 彬

上架指导：计算机科学及应用

ISBN 978-7-111-26524-5

9 787111 265245

定价：39.00 元